Bioinformatics, 37(18), 2021, 2803–2810
doi: 10.1093/bioinformatics/btab213
Advance Access Publication Date: 5 April 2021
Original Paper

Genome analysis

CoCoNet: an efficient deep learning tool for viral metagenome binning

Cédric G. Arisdakessian [1], Olivia D. Nigro[2], Grieg F. Steward[3], Guylaine Poisson[1] and Mahdi Belcaid[1,4,*]

[1]Department of Information and Computer Sciences, University of Hawai'i at Mānoa, Honolulu, HI 96822, USA, [2]Department of Natural Science, Hawai'i Pacific University, Honolulu, HI 96813, USA, [3]Department of Oceanography, University of Hawai'i at Mānoa, Honolulu, HI 96822, USA and [4]Hawai'i Institute of Marine Biology, University of Hawai'i at Mānoa, Honolulu, HI 96816, USA

*To whom correspondence should be addressed.

Associate Editor: Pier Luigi Martelli

Received on June 18, 2020; revised on March 24, 2021; editorial decision on March 25, 2021; accepted on April 2, 2021

Abstract

Motivation: Metagenomic approaches hold the potential to characterize microbial communities and unravel the intricate link between the microbiome and biological processes. Assembly is one of the most critical steps in metagenomics experiments. It consists of transforming overlapping DNA sequencing reads into sufficiently accurate representations of the community's genomes. This process is computationally difficult and commonly results in genomes fragmented across many contigs. Computational binning methods are used to mitigate fragmentation by partitioning contigs based on their sequence composition, abundance or chromosome organization into bins representing the community's genomes. Existing binning methods have been principally tuned for bacterial genomes and do not perform favorably on viral metagenomes.

Results: We propose Composition and Coverage Network (CoCoNet), a new binning method for viral metagenomes that leverages the flexibility and the effectiveness of deep learning to model the co-occurrence of contigs belonging to the same viral genome and provide a rigorous framework for binning viral contigs. Our results show that CoCoNet substantially outperforms existing binning methods on viral datasets.

Availability and implementation: CoCoNet was implemented in Python and is available for download on PyPi (https://pypi.org/). The source code is hosted on GitHub at https://github.com/Puumanamana/CoCoNet and the documentation is available at https://coconet.readthedocs.io/en/latest/index.html. CoCoNet does not require extensive resources to run. For example, binning 100k contigs took about 4 h on 10 Intel CPU Cores (2.4 GHz), with a memory peak at 27 GB (see Supplementary Fig. S9). To process a large dataset, CoCoNet may need to be run on a high RAM capacity server. Such servers are typically available in high-performance or cloud computing settings.

Contact: mahdi@hawaii.edu

Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

Shotgun metagenomics plays a critical role in investigating the composition and function of microbial communities in diverse environments, from the human gut (Tyagi *et al.*, 2019; Xie *et al.*, 2016) to the deep sea (Angly *et al.*, 2006; Hurwitz and Sullivan, 2013). Metagenomic assembly is particularly challenging for virome data (Sutton *et al.*, 2019). Viral shotgun assemblies are commonly plagued by short contigs, leading to poor characterization of the underlying species diversity, richness (García-López *et al.*, 2015) and functional capacity (Vázquez-Castellanos *et al.*, 2014). Binning contigs arising from a single species is a process routinely used to compensate for incomplete microbial metagenome assemblies. In the presence of reference genome sequences, binning is trivial and can be achieved by clustering contigs that align with high-confidence against the same reference genome. However, in samples containing species with unsequenced genomes, binning is more complicated since contigs need to be clustered *de novo*. While binning microbial contigs has seen significant advances recently, few programs have considered the unique set of challenges associated with viruses.

Existing methods for binning contigs can be divided into three classes: (i) methods based on sequence composition (Strous *et al.*, 2012), (ii) methods based on sequencing coverage correlation (Imelfort *et al.*, 2014) and (iii) a combination of both (Alneberg *et al.*, 2014; Kang *et al.*, 2019; Popic *et al.*, 2017). The first class of methods leverages the fact that bacterial species have predominantly

different distributions of *k*-mers, or words of size *k*, and use that evidence to bin together contigs with similar *k*-mer distributions. This approach works best on species with sufficiently divergent *k*-mer distributions but is less effective for resolving closely related species for which the *k*-mer distributions may be indistinguishable, particularly over short contigs. The second class of methods uses the sequencing coverage, i.e. the number of reads aligning to each contig, and reports contigs that consistently share similar coverage values across multiple samples as belonging to the same genome. This approach is accurate but works best when many samples are available (Popic *et al.*, 2017). The third class of methods leverages both *k*-mer and coverage profiles, typically using statistical models, to infer clusters. This class of methods combines the advantages of composition- and coverage-based solutions but is computationally more complex.

Here, we introduce Composition and Coverage Network (CoCoNet), a new method that leverages deep learning to model the *k*-mer composition and the coverage for binning contigs assembled from viral metagenomic data. Specifically, our method uses a neural network trained using contigs' subsequences, or fragments, to learn a flexible function for predicting the probability that any pair of contigs originated from the same genome. These probabilities are subsequently combined to infer bins representing the genomes of species present in the sequenced samples. Our approach was specifically optimized for viral metagenomes with large diversity, such as those found in environmental samples (e.g. oceans, soil, etc.). Such samples require sophisticated modeling methods that account for several sources of bias (Parras-Moltó *et al.*, 2018; Sutton *et al.*, 2019). We tested CoCoNet on both simulated and experimental viral metagenome data, and our results show that CoCoNet outperforms existing tools optimized for binning bacterial data. The CoCoNet source code and documentation are available at: https://github.com/Puumanamana/CoCoNet.

2 Materials and methods

2.1 The CoCoNet algorithm

We developed CoCoNet, a *Co*mposition and *Co*verage *Net*work that uses deep learning in conjunction with clustering to bin assembly contigs into homogeneous clusters representing the species present in the samples. Our approach works in two phases. The first phase trains a deep neural network to estimate the probability that two fragments belong to the same genome, given their composition and coverage information. The second phase uses a computationally tractable heuristic to bin the contigs using the co-occurrence probabilities inferred in the previous phase.

2.1.1 Preprocessing

To account for differences in contig lengths when extracting composition and coverage features, we divide contigs longer than 2048 bp into regularly spaced fragments of length 1024 bp (step: 128 bp).

Composition feature: We compute the composition feature by summing the number of occurrences of each *k*-mer, where $k = 4$, on each fragment's forward and reverse strands. This makes the distribution invariant to the reverse complement, therefore accounting for missing strand orientation information. Because this transformation induces *k*-mer redundancies (e.g. observing ACCG is the same as observing its reverse complement, CGGT), we are left with 136 unique *k*-mers.

Coverage feature: The coverage feature is computed by aligning the raw reads against the fragments, and by counting the number of reads aligning at each position of the fragment. To reduce artifacts due to erroneous read mapping and mis-assembly, we filtered the alignments to discard those that were either partial (represented 50% or less of the query's length), had a quality score lower than 30, or where one of the read pairs was unmapped (SAMflag = 3596). We also removed reads that had more than one high-quality alignment or PCR/optical duplicates. We excluded contigs occurring in only one sample (prevalence < 2) since our model does not extract relevant co-occurrence information from one sample. Finally, we smoothed the coverage using an averaging window of size 64 and sub-sampled the resulting vector every 32 bases to minimize computation and data storage requirements. These steps yielded 31 coverage values for each fragment in each sample.

Filtering complete genomes: Complete genomes were used in the training but were not included in the binning. We consider a contig to be complete if it contained Direct Terminal Repeats (Casjens and Gilcrease, 2009). Specifically, we flagged a contig as complete if its first and last 300 bp aligned over at least 10 bp with at least 95% identity.

2.1.2 Deep learning model

This section describes how we train a neural network *de novo* to learn to predict the probability that any pair of fragments belong to the same genome.

Training and test sets' construction: We divide the initial set of contigs into 90% training and 10% testing. Both sets contain positive and negative examples(see Fig. 1a). The positive examples are pairs of fragments from the same contig. They teach the model the similarities in fragments from the same genome. The negative examples are pairs of fragments from distinct contigs. They teach the model the differences between fragments from different genomes. When selecting positive training instances, we try to maximize the distance between the fragments to avoid trivial examples where fragments are similar due to their overlap. We generate the same number of positive examples from each contig in the training data to avoid biasing the training set in favor of long contigs. Using the approach described above, some mislabeling can occur with negative examples since different contigs can belong to the same bin. However, we show that the proportion of mislabeled pairs should be negligible for diverse metagenomes (see Supplementary Methods S1). Other studies have also shown that neural networks are robust to label noise and mislabeling, which can be mitigated with large batch sizes (Rolnick *et al.*, 2017).

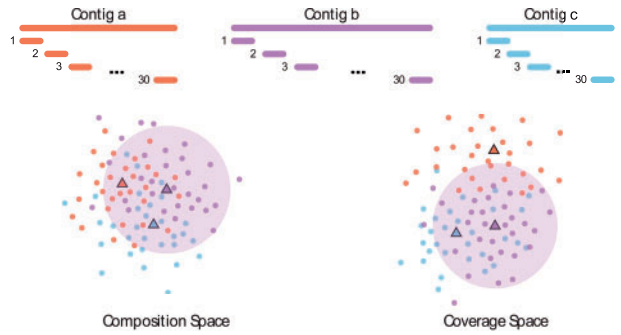

Fig. 1. Neural network training: (**a**) Training set construction: Each contig is split into 1024 bp fragments spaced with a 128 bp step. The training set is composed of two classes, positive (respectively negative) composed of fragment pairs from the same (respectively different) contigs. The positive class is constructed by extracting a constant number of contig pairs from each contig. For a given contig, we pick the most distant pairs. The negative class is composed of random fragments from random contig pairs. (**b**) Deep learning model: The neural network computes the fragments co-binning probability. First, a latent representation is computed for each composition and coverage features separately by sharing the weights of the network between the two inputs. A 64-neurons dense layer is used to process the composition vectors and a 1D convolution layer followed by a shared 64-neurons dense layer to process the coverage vectors. For a pair of fragments, the composition and the coverage features are merged with a siamese layer composed of a 32-neurons dense layer, followed by an element-wise maximum. Two 1-neuron layers are used to compute the probabilities of the observed composition and coverage values. The latent representations of the composition and coverage are also used to compute the probability of the combined coverage and composition. This is done by concatenating the latent feature representations (32-neural layer) and running them through a dense layer with 32-neurons, followed by a layer with 1-neuron and a sigmoid activation

Network architecture and training procedure: The network processes the composition and coverage inputs in three main steps (See Fig. 1b).

In the first step, we process both composition vectors separately using two 64-neurons dense layers that share the same weights. Similarly, we process both coverage vectors using two pairs of layers (1D convolution layer with 16 filters, a kernel_size = 4 and a stride = 2, and a 32-neurons dense layer) that share the same weights. Since the network outputs a probability of the two fragments belonging to the same genomes, the output should be symmetrical, i.e. $P(\text{frag}_1, \text{frag}_2) = P(\text{frag}_2, \text{frag}_1)$. The merging layer enforces this symmetricity using Siamese networks (Bromley *et al.*, 1993). In short, given two vectors x_1 and x_2 and a dense layer D, we compute both $D(x_1, x_2)$ and $D(x_2, x_1)$, and return the element-wise maximum between the two outputs. In the second step, the information from the four original inputs is aggregated into two vectors of size 64, representing the composition and coverage (Fig. 1b). Finally, the third step computes a composition probability, coverage probability and a combined probability that two fragments originated in the same genome.

We train the network using a batch size of 256 and a single epoch. We use the Adam (Kingma and Ba, 2014) optimizer with a learning rate of 10^{-3} and compute the overall loss as the weighted sum of the three binary cross-entropy (BCE) losses:

$$\text{Loss} = \text{BCE}(\text{composition}) + \text{BCE}(\text{coverage}) + 2 \cdot \text{BCE}(\text{combined})$$

The network accuracy is evaluated on the test data every 400 batches. The training stops if the test loss does not improve for consecutive 5 test batches.

2.1.3 Clustering

We bin the contigs using a clustering graph $G(v, e)$, where the nodes v represent the contigs and the edges, e link contigs belonging to the same bin. The edges computation, the comparison policy and the graph clustering steps are described in what follows.

1. Edges computation: During the clustering stage, we split each contig into 30 regularly spaced and maximally distant fragments of length 1024. To compare 2 contigs v_i and v_j, we compute the 900 probabilities that any fragment from v_i overlaps with fragments in v_j. We sum these probabilities to yield an expected number of hits between the contigs, which lies between 0 and 900:

$$\#\text{expected\_hits}(\text{v}_\text{i}, \text{v}_\text{j}) = \sum_{\text{l}=1}^{\text{m}} \sum_{\text{k}=1}^{\text{m}} \text{P}(\text{v}_\text{i}^\text{l}, \text{v}_\text{j}^\text{k}) \quad (1)$$

where v_i^l represents the l^{th} fragment in contig v_i and m is the number of fragments per contig (default is 30).

We subsequently assign an edge in G between contigs v_i and v_j if the number of expected hits is higher than a given threshold θ (by default, 80% of the maximum possible value of expected_hits, m^2).

2. Comparison policy: Given the large number of possible contig pairs in a dataset (about 0.5×10^9 comparisons for 30k contigs), we use a heuristic to evaluate only pairs that are more likely to belong to the same genome. In this heuristic, we hypothesize that for two contigs v_i and v_j, $\#\text{expected\_hits}(\text{v}_\text{i}, \text{v}_\text{j}) > \theta$ when v_i and v_j's fragments are close in both the composition and coverage spaces (see Supplementary Fig. S4). Given the high-dimensionality of the original features, we choose to investigate the spatial proximity of fragments in their latent representation space. We compute the latent representation of each contig in the composition and coverage spaces using the trained neural network and use that information to subsequently represent each contig in latent space as a ball centered at its fragments' center and with a radius R defined as:

$$R = \text{percentile}(\{||f - c_f||_2, f \in \text{fragments}\}, 90),$$

where $||f - c_f||_2$ is the euclidean distance of a fragment f to its center of mass c_f.

We initially limit the comparison of a query contig to its 250 closest neighbors, which we define as the contigs whose center of mass falls within the query's ball (See Fig. 2). Further, our approach emphasizes new comparisons rather than ones previously examined. Specifically, wherever contig v_j is selected for comparison to v_i, then contig v_i will not be selected for comparison to v_j. Since the radius is the same for all contigs, shorter contigs are not penalized due to the fact that their fragments are less variable. The number of neighbors was set to 250 after hyperparameter optimization. Our tests indicated no improvement in the results when considering more neighbors (see Supplementary Table S3).

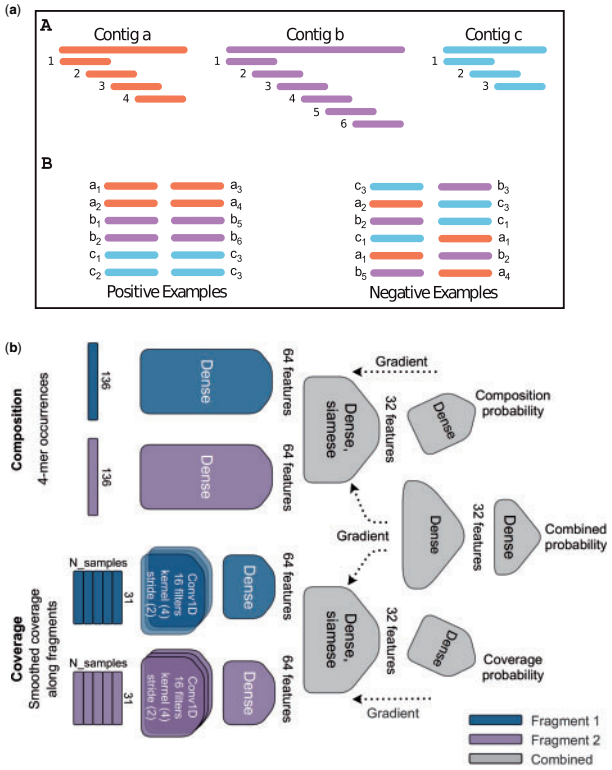

Fig. 2. Clustering method: We split each contig in 30 regularly spaced and maximally distant fragments and used the deep neural network to compute the latent representation for each fragment's composition and coverage features. Considering each feature type separately, we draw for each contig a ball centered at the center of mass of its fragments' latent representations, and with a radius equal to the 90th percentile of all distances between each fragments and their respective center of mass. We compare a contig to the 250 closest contigs whose center lies within the balls derived from both feature types. The bins are finally determined using the Leiden clustering algorithm

3. Graph clustering: We use the Leiden algorithm (Traag *et al.*, 2019) to identify the bins in the graph representing the species. This clustering algorithm infers its clusters by optimizing the Constant Potts Model (CPM) quality function H defined as:

$$H = \sum_{c \in \text{clusters}} \text{edges}(c) - \gamma \cdot \binom{\text{vertices}(c)}{2}$$

where edges$(\cdot)$ (respectively vertices$(\cdot)$) counts the number of edges (respectively vertices) in a given cluster. The parameter γ controls how dense clusters are in terms of edges. Initially, each edge belongs to a different cluster. Then, the Leiden algorithm repeatedly (i) moves nodes between clusters to improve H, (ii) refines the clusters into sub-communities and (iii) aggregates all the edges of a cluster into a single aggregated network. The algorithm stops when H cannot be improved further.

Since the clustering graph has potentially missing edges resulting from our heuristic, we perform the clustering in two steps. First, we run the Leiden algorithm with a small resolution parameter ($\gamma = 0.3$) to delineate unpolished clusters. Then, within each cluster, we fill the adjacency matrix with the remaining comparisons and re-

run the Leiden clustering algorithm within each cluster with a higher resolution parameter ($\gamma = 0.4$). The CoCoNet implementation includes a second community detection algorithm, spectral clustering (Newman, 2006), which is ideal when the number of bins is known.

2.2 Metrics

2.2.1 Classification metrics

We measured the network's ability to classify contig pairs using the accuracy, the Area Under the Receiver Operating Characteristics (ROC) Curve (AUC) and the F1 score. These measures are defined as follows:

- The accuracy is the proportion of good predictions (either true positives or true negatives) among all predictions.
- AUC: Area under the ROC curve defined as [TPR(thresh), FPR(thresh)] at various classification thresholds.
- The F1 score is a summary value that takes into account both precision and recall. It is defined as:

$$F_1 = 2 \cdot \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} = \frac{2 \cdot \text{TP}}{2 \cdot \text{TP} + \text{FN} + \text{FP}}$$

2.2.2 Clustering metrics

To compare the binning predicted by CoCoNet, CONCOCT (Alneberg *et al.*, 2014) and Metabat2 (Kang *et al.*, 2019) against the ground truth, we used the Adjusted Rand Index (ARI), the homogeneity and the completeness. These measures are defined as follows:

- The ARI (Hubert and Arabie, 1985) is the ratio of all pairs that are assigned correctly as belonging either together or not together, among all possible pairs. It is adjusted for chance.
- The homogeneity measures whether clusters contain only members of a single class. It is derived from the ratio of $H(C|K)$, the conditional entropy of the classes given the cluster assignments and the entropy of the cluster assignments.

$$\text{homogeneity} = 1 - \frac{H(C|K)}{H(C)}$$

- The completeness measures whether all members of a given class are assigned to the same cluster. It is derived from the ratio of $H(K|C)$, the conditional entropy of the cluster assignments given the classes and the entropy of the cluster assignments.

$$\text{completeness} = 1 - \frac{H(K|C)}{H(K)}$$

2.2.3 CheckV analysis

We used CheckV v0.7.0 (Nayfach *et al.*, 2020) to analyze the quality of the bins resulting from the Station ALOHA dataset. We concatenated the contigs in each bin into an artificial contig as required by CheckV. We also ensured that any Direct Terminal Repeats found in the bin occur at the ends of the resulting construct. We reported our results using CheckV's categories (Complete, high-quality, medium-quality, low-quality, undetermined).

2.3 Simulations and experimental datasets

2.3.1 Simulations

Reference genomes were downloaded from the NCBI RefSeq viral database (O'Leary *et al.*, 2016) (ftp://ftp.ncbi.nlm.nih.gov/refseq/release/viral/; dataset version: September 12, 2019). This dataset contains 13 274 viruses, 9316 of which had a length greater than 3 kb. The dataset was first preprocessed to convert ambiguous IUPAC codes into A, C, G, T by randomly sampling among the possible values of each ambiguous code. Unresolved nucleotides, or Ns, were removed from the kmer counts to avoid biasing the computation. Finally, genomes shorter than 3 kb were discarded from the simulations as they were too small to yield at least two contigs of 2048 bp.

We used the preprocessed viral dataset with CAMISIM (Fritz *et al.*, 2019) to simulate four synthetic metagenomes. Each of the synthetic metagenomes had either 4 or 10 samples and a coverage of either 3× or 10×. For each simulation, we randomly sampled 500 or 2000 genomes and used a log-normal distribution ($\mu = 1$, $\sigma = 3$) to simulate each genome's relative abundance. Each of the four experimental conditions was repeated ten times to account for sampling variability. This resulted in a total of 80 simulations.

We simulated the sequencing by fragmenting each genome into chunks with a mean length of 400 bp and a standard deviation of 10 bp. Each subsequence was then used to simulate paired-end reads with errors introduced using Illumina's MiSeq error profile. Contigs were inferred deterministically using CAMISIM's simulated reads' location in the reference genome. In this approach, which CAMISIM refers to a gold standard assembly, a contig is simply a consensus sequence of the reads simulated from the region it spans.

2.3.2 Experimental datasets

We used three viral metagenomes from Beaulaurier *et al.* The data was collected from Station ALOHA in the North Pacific Subtropical Gyre and sequenced with both Oxford Nanopore Technologies (ONT) and Illumina technologies. Raw Illumina reads were downloaded from SRA (SRR10378148, SRR8811962, SRR8811963) and trimmed with Fastp (Chen *et al.*, 2018): reads were truncated when the quality on a sliding window of length 10 bp dropped below 25. Leading and trailing bases with quality below 5 were also trimmed. All reads shorter than 20 bp were filtered out. Trimmed reads were assembled with MetaSPAdes (Nurk *et al.*, 2017). The resulting assembly contained 59 027 contigs longer than 2048 bp. Sequencing reads were aligned against the contigs using the bwa-mem algorithm (Li, 2013) with default parameters. The alignments were filtered using *pysam* (Anders *et al.*, 2015) to drop those that were partial (less than 50% of the query), had a low quality (30 or less) or contained an unmapped mate-pair (SAM flag = 3596). We also filtered out alignments where reads had multiple alignments or PCR/optical duplicates. Finally, we removed paired alignments with mapping distance less than 200 or greater than 500 bp (see template length distribution in Supplementary Fig. S1). The number of contigs filtered at each step is provided in Supplementary Table S1. After filtering, 42% of the contigs in each sample had a coverage above 1×.

These genomes were assembled and polished using the ONT reads as described in Beaulaurier *et al.* (2020) (accession: PRJNA529454). This step resulted in 567, 96 and 1217 high-quality draft genomes in the respective samples. We pooled all the contigs and dereplicated them using Cluster-Genomes (Roux and Bolduc, 2009) using the default parameter values (minimum nucleotide identity = 95%, min coverage = 80%). Only 1322 unique contigs remained after the dereplication.

Given that ONT and Illumina data were sequenced from the same biological samples, we aligned the shorter Illumina contigs against the ONT reference assembly to obtain the ground truth needed to verify our predictions. Thus, two contigs that align against the same genome reference must bin together in the ground truth solution. Conversely, pairs of contigs aligning to different genomes must appear in separate bins in the ground truth solution. We used minimap2 (Li, 2018) to align the Illumina contigs against the ONT references and filter out potentially erroneous alignments if their minimum nucleotide identity was less than 95% (same threshold as Cluster-Genomes). We also discarded alignments where the Illumina contigs were longer than the ONT reference. Such alignments don't convey information conducive to validating our binning predictions.

Table 1. For each experimental condition, the means and standard deviations of ten replicates are shown

Dataset	Number of genomes	Coverage	Number of samples	Average bin size	Average prevalence	Number of contigs (>2kb)
Sim-1	500	3×	4	3.16 ± 0.74	2.25 ± 0.07	678 ± 165
Sim-2	500	3×	15	1.89 ± 0.15	6.30 ± 0.37	893 ± 70
Sim-3	500	10×	4	2.54 ± 0.39	2.61 ± 0.12	838 ± 173
Sim-4	500	10×	15	1.17 ± 0.12	8.54 ± 0.32	582 ± 57
Sim-5	2000	3×	4	3.25 ± 0.32	2.18 ± 0.07	2554 ± 195
Sim-6	2000	3×	15	2.07 ± 0.09	5.97 ± 0.10	3871 ± 163
Sim-7	2000	10×	4	2.48 ± 0.08	2.56 ± 0.08	3209 ± 93
Sim-8	2000	10×	15	1.27 ± 0.14	7.96 ± 0.29	2520 ± 271
SA	>1300	6.6× ±2.2	3	N/A	1.51 ± 0.61	59 027

Note: The number of bins for the Station ALOHA dataset is unknown but is estimated to be at least the number of curated contigs assembled from the Oxford Nanopore long reads. The table provides the number of contigs in each simulation, the average bin size and the average prevalence (number of samples in which a contig occurs). The variance for each of these values across the ten replicate simulations is provided after the '±' symbol. SA refers to the experimental dataset 'Station ALOHA'.

2.4 Other algorithms

We used CONCOCT's latest version 1.1.0 and set the maximum number of genomes to 2000 (maximum number of species in our simulations). We ran Metabat2 v2.15 with the default parameters.

3 Results

3.1 Viral datasets

3.1.1 Simulated datasets

We used a simulated dataset constructed using the NCBI's complete set of viral genomes (see Methods for more details). 38% of the viruses in the database belong to the *Caudovirales* order, a majority of which are of the *Siphoviridae* family (See Supplementary Fig. S7). As expected, the simulations were less fragmented as the number of samples or the coverage increased. For example, with 4 samples and 500 genomes, increasing the coverage from 4× to 10× decreases the number of contigs by 26% (23 258 to 17 303). Similarly, with 4× coverage and 500 genomes, increasing the samples from 3 to 15 leads to an 80% decrease in the number of contigs (see Supplementary Table S1). Table 1 displays some relevant properties of the simulated dataset in terms of average bin size, prevalence (number of samples in which a contig occurs) and number of contigs longer than 2 kb. The bin size distribution for each setting is available in Supplementary Figure S8.

3.1.2 Experimental dataset 'Station ALOHA'

A total of 2332 Illumina contigs (longer than 2048 bp) mapped against 988 ONT references with 95% similarity or higher. These alignments are valuable for providing the binning ground truth, which we relied on in our tests.

3.1.3 Effect of fragment length on composition separation

We compared the distributions of cosine distances between the *k*-mer profiles of pairs of subsequences within versus across 9316 publicly available viral genomes. Differences between intra- versus inter-genome distance distributions were significant (t-test) at all fragment lengths tested although the distributions were very similar for the smallest fragment sizes (256 bp). The cosine distances progressively diverged (range 0.10–0.48) as a function of fragment length (Fig. 3), suggesting that *k*-mer composition may represent a robust feature for binning when computed on fragments of at least 1024 bp. The high prevalence of viral genetic recombinations (Lai, 1992) and host to virus horizontal transfers (Gilbert *et al.*, 2016) render binning using *k*-mer composition alone impractical. This point is evidenced by the overlap between the across- and within-species distributions in Figure 3. Other binning methods have suggested that coverage can facilitate recovery of more complete bins (Alneberg *et al.*, 2014). As such, it is crucial to include that information to enhance the signal conveyed by the *k*-mer composition.

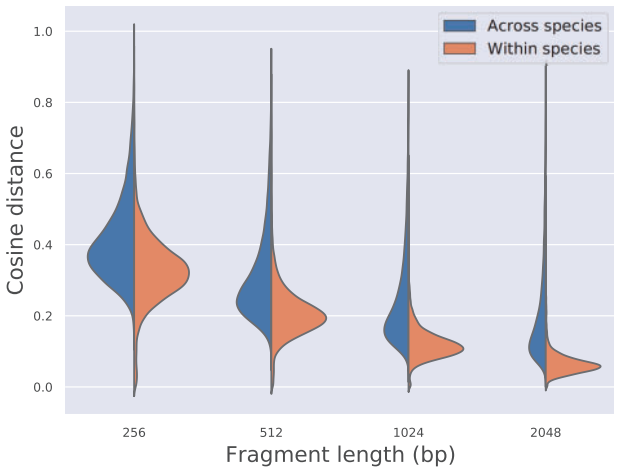

Fig. 3. Distribution of *k*-mer distances: The *k*-mer cosine distance between fragments from the same genome (blue) and different genomes (orange) for fragments of length 256, 512, 1024 and 2048 bp. The variance of the cosine distance within and across species decreases gradually as the fragment length increases. The overlap between the two distributions is much smaller for fragments of 1024 bp or longer. This suggests that the best power for distinguishing within and across species requires fragments of at least 1024 bp

3.2 CoCoNet performance on simulated viral data

For each simulated dataset, we trained the deep neural network using the 4-mer frequencies and coverage vectors as inputs. We measured the network learning performance using the accuracy, the F1 score and the Area Under the ROC Curve (see methods for details about these metrics).

Overall, the accuracy, the F1-score and the AUC are all consistently above 95% (Fig. 4). All of the metrics increase with the number of samples. Indeed, the larger the number of samples, the easier and more accurate it becomes for the model to learn the expected variability in coverage across contigs from the same genome. The computed quality metrics are also correlated with the sequencing depth since a higher coverage can lead to fewer contigs, more complete genomes and less coverage variance between contigs of the same genome. We also observe a slight decrease in the variance and improved metrics when 2000 genomes were included in the simulations. This larger number of species allows for more diversity in training examples and more possibilities for generalization. The total counts for true positives, true negatives, false positives, false negatives show a similar trend (see Supplementary Fig. S2).

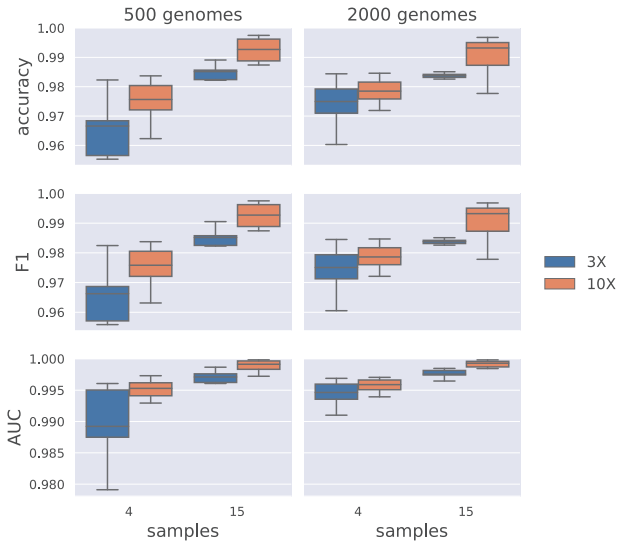

Fig. 4. Neural network performance on simulated data for varying number of genomes (500, 2000), samples (4, 15) and coverage (blue: 3, orange: 10). The distribution of the scores is shown for each of the 3 metrics (AUC, Accuracy, F1 score)

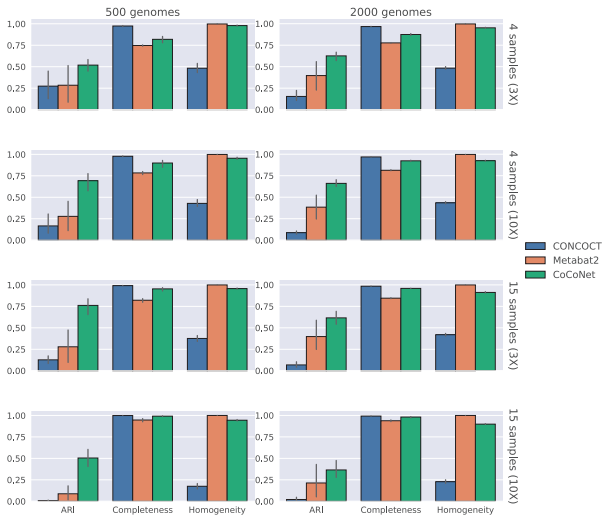

Fig. 5. Clustering performance on simulated data for varying number of genomes (500, 2000), samples (4, 15) and coverage (3, 10). For each of the 3 metrics (ARI, Completeness, Homogeneity), the distribution of the scores is provided for each method (blue: CoCoNet, orange: CONCOCT, green: Metabat2)

The binning performances of CoCoNet, CONCOCT and Metabat2 were compared for each of the simulated datasets using the ARI (Hubert and Arabie, 1985), the completeness and the homogeneity (Fig. 5). The completeness measures true positives, or events where contigs were correctly binned together, while homogeneity measures true negatives, or events where contigs were correctly assigned to separate bins. ARI is a more general metric that captures both types of events (see methods for more details). CoCoNet accurately reconstructs, on average, 14% of the bins that contain at least two contig, compared to 2.1% and 2.0% for Metabat2 and CONCOCT, respectively. Overall, CONCOCT assigned all the contigs to a small number of bins, yielding high levels of false positives. For instance, the maximum number of bins observed was 70 and occurred for a coverage of 3×, 15 samples and 2000 genomes. On the other hand, Metabat2 creates a small number of homogeneous, non-singleton bins (low number of false positives). In summary, CONCOCT has low homogeneity (median = 0.418) and high completeness (median = 0.984) while Metabat2, has high homogeneity

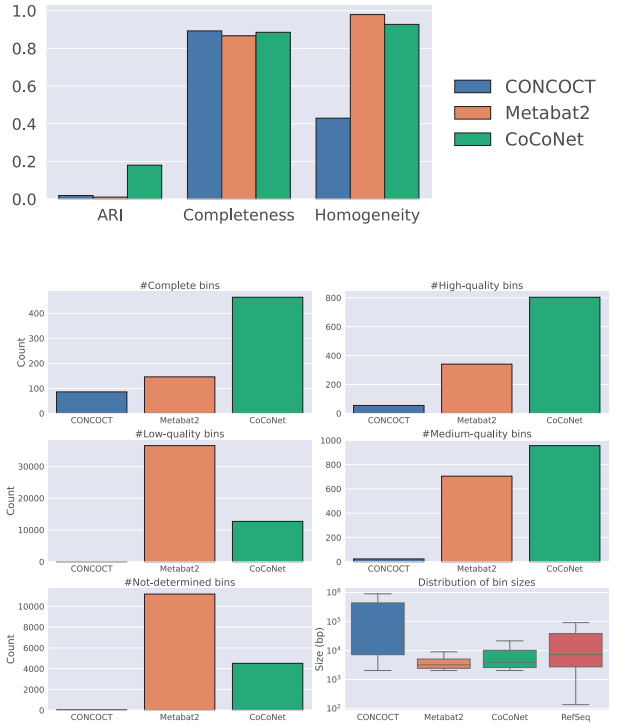

Fig. 6. (Top) Clustering performance on Station ALOHA data. The scores are provided for each of the 3 metrics (ARI, Completeness, Homogeneity) and each method (blue: CoCoNet, orange: CONCOCT, green: Metabat2). (Bottom) Bin classification by CheckV. Each barplot corresponds to the number of contigs in the given category. The last facet (boxplot) is the distribution of bin sizes for each method. The genome length distribution of the viral RefSeq database is shown as a reference

(median = 0.999) and lower completeness (median = 0.817). In contrast, CoCoNet achieves high scores on both metrics, with a median homogeneity and completeness values of 0.944 and 0.942, respectively. Since the completeness measures whether contigs from the same virus are binned together, having a lower number of clusters artificially inflates this metric, i.e. decreasing the number of clusters can only increase the chance of grouping contigs. The ARI is a more generic metric since it measures both correct and erroneous binning events by considering all pairs of contigs. The ARI scores for bins produced by CONCOCT and Metabat2 were low due to their high false positive and false negative rates, respectively. Specifically, CONCOCT and Metabat2 have median ARI scores of 0.0794 and 0.200, whereas CoCoNet achieves a median score of 0.617.

Altogether, CoCoNet outperforms both CONCOCT and Metabat2 on the simulated data as it yields more bins containing contigs from the same genome and fewer bins containing contigs from different genomes.

3.3 CoCoNet accurately identifies contigs in the experimental dataset 'Station ALOHA'

We processed all Illumina assembly contigs using CoCoNet, CONCOCT and Metabat2, and computed the ARI, completeness and homogeneity scores using only the subset of contigs mapping against the draft genomes. Figure 6 compares the scores of the three metrics (ARI, Homogeneity and completeness) for all 3 methods.

Similar to the results observed in the simulation, CONCOCT generates only 61 bins in total. Nineteen (19) of these bins are homogeneous, and only 9 were non-singleton. Metabat2 generated 2212 bins, with 2196 singletons, one homogeneous but partial and 16 with erroneously merged contigs. In contrast, CoCoNet generates 1452 bins, 1177 were homogeneous, and 127 were homogeneous and non-singletons (see Supplementary Table S2).

Metabat2 generated many bins, resulting in a very high homogeneity (0.979) and completeness (0.869). However, the ARI value

was low (0.00108) since only one non-singleton bin was correct but small (2 contigs). CONCOCT achieved a slightly higher completeness score (0.893) but a low homogeneity (0.191) and ARI (0.0103). In contrast, CoCoNet results in a completeness value on par with CONCOCT and Metabat2 (0.885) but has substantially higher homogeneity (0.927) than CONCOCT and an ARI (0.180) an order of magnitude higher than the two other methods.

To further evaluate bin quality, we used CheckV to classify bins into five categories (complete, high-quality, medium-quality, low-quality and not-determined). CheckV's results mirror the ARI scores, with 464 complete bins and 804 high-quality bins for CoCoNet, 146 complete bins and 341 high-quality bins for Metabat2 and 86 complete bins and 55 high-quality bins for CONCOCT (see Fig. 6). We also compared the bin size distribution of each method with the RefSeq database to provide another qualitative assessment of bin completeness and contamination. Indeed, if bins are complete, then their size (in bp) should be similar to what can be found in reference viral databases. If bins are significantly larger, it could be a sign of contamination. For example, CONCOCT's median bin size (39 194) is much higher than in RefSeq (12 155 for genomes longer than 3 kb, 7253 overall). CoCoNet and Metabat2 are more comparable to RefSeq with medians equal to 3926 and 3204, respectively.

4 Discussion

Our results show that CoCoNet performs substantially better than existing methods for binning contigs assembled from viral metagenomes. Using CAMISIM, we simulated 80 datasets with varying number of bins, coverage and number of samples. Overall, CoCoNet retrieved the correct bins with fewer errors compared to either CONCOCT or Metabat2. On a dataset that includes a high-quality ONT assembly and Illumina shotgun sequencing reads of the same samples, we showed that CoCoNet was able to retrieve substantially more correct bins than CONCOCT and Metabat2. This is particularly noteworthy since the dataset contained only three samples.

Unlike binning algorithms that summarize the coverage across the complete contig using one or two values (e.g. mean and standard deviation in the case of Metabat2), CoCoNet learns to model the inherent coverage variability within samples by considering the coverage in a sliding window. This distinction is critical in viral metagenomes where the DNA amplification methods used to increase the typically low input material (Parras-Moltó *et al.*, 2018; Sutton *et al.*, 2019) can yield uneven coverage depths (Karlsson *et al.*, 2013; Rosseel *et al.*, 2013) and, therefore, confound methods that rely on summary statistics. We further highlight the importance of coverage variability in Supplementary Figure S3, which shows that the neural network's AUC generally decreases when the coverage variability is subtracted (i.e. coverage is set to the mean value).

CoCoNet default behavior is to drop contigs shorter than 2048. CONCOCT, and Metabat2 also have minimum contig length thresholds. For example, Metabat2 uses a minimum contig length of 2.5 kb in its model and relies on an ad-hoc heuristic to incorporate short contigs into the inferred bins. Here, we have decided to drop contigs shorter than 2048 bases for two reasons. First, shorter contigs are difficult to classify (see Supplementary Fig. S5 in Supplementary Material) and their inclusion in the computation is unlikely to be without impact on the false-positive rate. Second, the more positive pairs we predict across two contigs (Figure 2B), the more likely we can assign those contigs to the same bin. To generate training instances from short contigs, we need either (i) allow the fragments to overlap more or (ii) take shorter fragments. The first solution results in training instances that are highly correlated. The second option results in training instances that suffer from high variance. In our tests, both solutions resulted in poor generalization power of the model.

We generate our negative examples by considering pairs of fragments that belong to different contigs. Our assumption here is that two randomly selected contigs are most likely to belong to separate rather than the same genome. Specifically, for large metagenomes such as those found in environmental samples (e.g. oceans, soil, etc.), we expect contigs pairs originating from different genomes to outnumber contig pairs originating from the same species by a few orders of magnitude. As such, label noise would only represent a negligible fraction of the negative examples generated (See Supplementary Methods S1). Studies have also shown that neural networks are robust to massive label noise, which can be mitigated with large batch sizes (Rolnick *et al.*, 2017). CoCoNet uses a batch size of 256, which is reasonably large.

The coverage variability resulting from amplification bias (Parras-Moltó *et al.*, 2018) or coverage heteroskedasticity, where the coverage's standard deviation varies along a contig (Hugerth and Andersson, 2017) are other sources of noise that can confound the binning. Deep learning models are also relatively robust to such sources of noise (D'Souza *et al.*, 2020). This makes CoCoNet more suitable for handling potential variability in sequencing coverage than alternative methods that summarize a contig's coverage as a single value.

CoCoNet's clustering approach is very conservative and emphasizes bin homogeneity. Three of CoCoNet's parameters can be adjusted to improve bin completeness at the cost of possibly decreased homogeneity. Those are (i) fragment length, (ii) the minimum number matches between two contigs connected by an edge in the contig-contig graph (θ) and (iii) the minimum edge density required for considering a cluster as a bin (γ). Decreasing the values of θ or γ (respectively 80% and 75% by default) decreases the binning stringency.

Similarly, increasing the fragment length can minimize the variance in the k-mer and coverage distributions between contigs of the same species and, consequently, improve completeness (see Fig. 3). Nevertheless, a longer fragment length can result in more contigs being assigned to singleton bins simply because they were not long enough to be processed. Naturally, decreasing the values of θ, γ, or the fragment length can result in more homogeneous but less complete bins (see Supplementary Fig. S6).

Finally, CoCoNet's accuracy can be bolstered by combining multiple related experiments. As was shown with the simulated dataset, CoCoNet's performance strongly increases as we increase the number of samples.

5 Conclusion

CoCoNet is a novel, deep learning-based approach to bin viral metagenome assemblies. Our strategy leverages deep neural networks' flexibility to learn the similarity in composition and coverage across co-occurring contigs. Contrary to other methods that rely on coverage and k-mer composition, our method models these features as distributions rather than summarizing them in a single statistic. Our results show that CoCoNet outperforms other tools on both simulated and real viral datasets. Our model is implemented in a self-contained easy to install Python program that requires reasonable computational resources to train the underlying deep neural network or use it to predict bins.

Funding

This work was supported by funding from the National Science Foundation Division of Ocean Sciences [1636402-Investigation of viruses and microbes circulating deep in the seafloor] and the Office of Integrative Activities [1557349-Ike Wai: Securing Hawaii's Water Future and 1736030–G2P in VOM: An experimental and analytical framework for genome to phenome connections in viruses of microbes].

Author Contributions

M.B. envisioned the project, C.A. implemented the project and conducted the analysis with the help of M.B. C.A., M.B., G.P., O.N. and G.S. discussed the analyses and the results and wrote the manuscript. All authors have read and agreed to the published version of the manuscript.

Conflict of Interest: none declared.

References

Alneberg,J. *et al.* (2014) Binning metagenomic contigs by coverage and composition. *Nat. Methods*, **11**, 1144–1146.

Anders,S. *et al.* (2015) Htseq-a python framework to work with high-throughput sequencing data. *Bioinformatics*, **31**, 166–169.

Angly,F. *et al.* (2006) The marine viromes of four oceanic regions. *PLoS Biol.*, **4**, e368.

Beaulaurier,J. *et al.* (2020) Assembly-free single-molecule sequencing recovers complete virus genomes from natural microbial communities. *Genome Res.*, **30**, 437–446.

Bromley,J. *et al.* (1993) Signature verification using a "siamese" time delay neural network. In *Proceedings of the 6th International Conference on Neural Information Processing Systems, NIPS'93*, pp. 737–744.

Casjens,S.R. and Gilcrease,E.B. (2009) Determining DNA packaging strategy by analysis of the termini of the chromosomes in tailed-bacteriophage virions. In: Clokie, M.R.J., Kropinski, A.M. (eds) *Bacteriophages*. Springer, Totowa, Humana Press, NJ, USA, pp. 91–111.

Chen,S. *et al.* (2018) fastp: an ultra-fast all-in-one fastq preprocessor. *Bioinformatics*, **34**, i884–i890.

D'Souza,S. *et al.* (2020) Machine learning in drug–target interaction prediction: current state and future directions. *Drug Discov. Today*, **25**, 748–756.

Fritz,A. *et al.* (2019) Camisim: simulating metagenomes and microbial communities. *Microbiome*, **7**, 1–12.

García-López,R. *et al.* (2015) Fragmentation and coverage variation in viral metagenome assemblies, and their effect in diversity calculations. *Front. Bioeng. Biotechnol.*, **3**, 141.

Gilbert,C. *et al.* (2016) Continuous influx of genetic material from host to virus populations. *PLoS Genet.*, **12**, e1005838.

Hubert,L. and Arabie,P. (1985) Comparing partitions. *J. Classif.*, **2**, 193–218.

Hugerth,L.W. and Andersson,A.F. (2017) Analysing microbial community composition through amplicon sequencing: from sampling to hypothesis testing. *Front. Microbiol.*, **8**, 1561.

Hurwitz,B.L. and Sullivan,M.B. (2013) The Pacific Ocean Virome (POV): a marine viral metagenomic dataset and associated protein clusters for quantitative viral ecology. *PLoS One*, **8**, e57355.

Imelfort,M. *et al.* (2014) GroopM: an automated tool for the recovery of population genomes from related metagenomes. *PeerJ*, **2**, e603.

Kang,D.D. *et al.* (2019) Metabat 2: an adaptive binning algorithm for robust and efficient genome reconstruction from metagenome assemblies. *PeerJ*, **7**, e7359.

Karlsson,O.E. *et al.* (2013) The effect of preprocessing by sequence-independent, single-primer amplification (SISPA) on metagenomic detection of viruses. *Biosecurity Bioterrorism Biodefense Strat. Pract. Sci.*, **11**, S227–S234.

Kingma,D.P. and Ba,J. (2014) Adam: a method for stochastic optimization. *arXiv Preprint arXiv, 1412, 6980*.

Lai,M. (1992) Genetic recombination in RNA viruses. In: Holland J.J. (ed). *Genetic Diversity of RNA Viruses*. Springer, Berlin, Heidelberg, pp. 21–32.

Li,H. (2013) Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM. *arXiv Preprint arXiv, 1303, 3997*.

Li,H. (2018) Minimap2: pairwise alignment for nucleotide sequences. *Bioinformatics*, **34**, 3094–3100.

Nayfach,S. Camargo,A.P., Schulz,F., Eloe-Fadrosh,E., Roux,S., and Kyrpides,N.C. (2020) Checkv: assessing the quality of metagenome-assembled viral genomes. *Nature Biotechnol., 1–8*.

Newman,M.E. (2006) Finding community structure in networks using the eigenvectors of matrices. *Phys. Rev. E*, **74**, 036104.

Nurk,S. *et al.* (2017) metaspades: a new versatile metagenomic assembler. *Genome Res.*, **27**, 824–834.

O'Leary,N.A. *et al.* (2016) Reference sequence (refseq) database at ncbi: current status, taxonomic expansion, and functional annotation. *Nucleic Acids Res.*, **44**, D733–D745.

Parras-Moltó,M. *et al.* (2018) Evaluation of bias induced by viral enrichment and random amplification protocols in metagenomic surveys of saliva DNA viruses. *Microbiome*, **6**, 119.

Popic,V. *et al.* (2017) GATTACA: lightweight metagenomic binning with compact indexing of kmer counts and minhash-based panel selection. *bioRxiv*, page 130997.

Rolnick,D. *et al.* (2017) Deep learning is robust to massive label noise. *arXiv Preprint arXiv, 1705, 10694*.

Rosseel,T. *et al.* (2013) The origin of biased sequence depth in sequence-independent nucleic acid amplification and optimization for efficient massive parallel sequencing. *PLoS One*, **8**, e76144.

Roux,S. and Bolduc,B. (2009) *ClusterGenomes*. doi: 10.17504/protocols.io.gwebxbe

Strous,M. *et al.* (2012) The binning of metagenomic contigs for microbial physiology of mixed cultures. *Front. Microbiol.*, **3**, 410.

Sutton,T.D. *et al.* (2019) Choice of assembly software has a critical impact on virome characterisation. *Microbiome*, **7**, 12.

Traag,V. *et al.* (2019) From Louvain to Leiden: guaranteeing well-connected communities. *Sci. Rep.*, **9**, 5233.

Tyagi,A. *et al.* (2019) Shotgun metagenomics offers novel insights into taxonomic compositions, metabolic pathways and antibiotic resistance genes in fish gut microbiome. *Arch. Microbiol.*, **201**, 295–303.

Vázquez-Castellanos,J.F. *et al.* (2014) Comparison of different assembly and annotation tools on analysis of simulated viral metagenomic communities in the gut. *BMC Genomics*, **15**, 37.

Xie,H. *et al.* (2016) Shotgun metagenomics of 250 adult twins reveals genetic and environmental impacts on the gut microbiome. *Cell Syst.*, **3**, 572–584.