Exploring neighborhoods in large metagenome assembly graphs reveals hidden sequence diversity — Source link

Genome Biology

METHOD Open Access

Exploring neighborhoods in large metagenome assembly graphs using spacegraphcats reveals hidden sequence diversity

C. Titus Brown[1]*, Dominik Moritz[2], Michael P. O'Brien[3], Felix Reidl[3], Taylor Reiter[1] and Blair D. Sullivan[3]*

*Correspondence:
ctbrown@ucdavis.edu;
sullivan@cs.utah.edu
[1]Department of Population Health and Reproduction, University of California Davis, Davis, USA
[3]Department of Computer Science, NC State University, Raleigh, USA
Full list of author information is available at the end of the article

Abstract
Genomes computationally inferred from large metagenomic data sets are often incomplete and may be missing functionally important content and strain variation. We introduce an information retrieval system for large metagenomic data sets that exploits the sparsity of DNA assembly graphs to efficiently extract subgraphs surrounding an inferred genome. We apply this system to recover missing content from genome bins and show that substantial genomic sequence variation is present in a real metagenome. Our software implementation is available at https://github.com/spacegraphcats/spacegraphcats under the 3-Clause BSD License.

Keywords: Metagenomics, Sequence assembly, Strain variation, Bounded expansion, Dominating set

Metagenomics is the analysis of microbial communities through shotgun DNA sequencing, which randomly samples many subsequences (reads) from the genomic DNA of each microbe present in the community [1].

A common problem in metagenomics is the reconstruction of individual microbial genomes from the mixture. Typically, this is done by first running an assembly algorithm that reconstructs longer linear regions based on a graph of the sampled subsequences [2], and then binning assembled contigs together using compositional features and gene content [3, 4]. These "metagenome-assembled genomes" are then analyzed for phylogenetic markers and metabolic function. In recent years, nearly 200,000 metagenome-assembled genomes have been published, dramatically expanding our view of microbial life [5–10].

Information present in shotgun metagenomes is often omitted from the binned genomes due to either a failure to assemble [11, 12] or a failure to bin. The underlying technical reasons for these failures include low coverage, high sequencing error, high strain variation, and/or sequences with insufficient compositional or coding signal. Recent work has particularly focused on the problem of strain confusion, in which high strain variation results in considerable fragmentation of assembled content in mock or synthetic communities [11, 12]; the extent and impact of strain confusion in real metagenomes is still unknown but potentially significant—metagenome-assembled genomes may be missing 20–40% of true content [13–15].

Associating unbinned metagenomic sequence to inferred bins or known genomes is technically challenging. Some approaches use mapping or *k*-mer baiting, in which assembled sequences are used to extract reads or contigs from a metagenome or graph [16–20]. These methods fail to recover genomic content that does not directly overlap with the query, such as large indels or novel genomic islands. Moreover, most assembly graphs undergo substantial heuristic error pruning and may not contain relevant content [11, 12]. Graph queries have shown promise for recovering sequence from regions that do not assemble well but are graph-proximal to the query [21, 22]. However, many graph query algorithms are NP-hard and hence computationally intractable in the general case; compounding the computational challenge, metagenome assembly graphs are frequently large, with millions of nodes, and require 10s to 100s of gigabytes of RAM for storage.

In this paper, we develop and implement a scalable graph query framework for extracting unbinned sequence from metagenome assembly graphs with millions of nodes. Crucially, we exploit the structural sparsity of compact De Bruijn assembly graphs in order to compute a succinct index data structure in linear time. Our initial investigations presented here focus on using this index to perform neighborhood queries in large assembly graphs to investigate genome binning and content recovery. This enables us to extract the genome of a novel bacterial species, recover missing sequence variation in amino acid sequences for genome bins, and identify missing content for metagenome-assembled genomes. Our query methods are assembly-free and avoid techniques that may discard strain information such as error correction. These algorithms are available in an open-source Python software package, spacegraphcats [23].

Results

Dominating sets enable efficient neighborhood queries in large assembly graphs

We designed and implemented [23] a set of algorithms for efficiently finding content in a metagenome that is close to a query as measured by distance in a compact De Bruijn graph (cDBG) representation of the sequencing data (Fig. 1). To accomplish this, we organize the cDBG into *pieces* around a set of *dominators* that are collectively close to all vertices. In this context, the *neighborhood* of a query is the union of all pieces it overlaps; to enable efficient search, we build an invertible index of the pieces.

We compute dominators so that the minimum distance from every vertex in the cDBG to some dominator is at most r (an *r-dominating set*) using Algorithm 1, which is based on the linear-time approximation algorithm given by Dvořák and Reidl [24]. Although finding a minimum r-dominating set is NP-hard [25–27] and an approximation factor below $\log n$ is probably impossible [26] in general graphs, our approach guarantees constant-factor approximations in linear running time by exploiting the fact that (compact) De Bruijn graphs have *bounded expansion*, a special type of sparsity [28]. Algorithm 1 first annotates the graph to determine the distances between all pairs of vertices at distance at most r (lines 1–3) and then uses these distances to ensure each

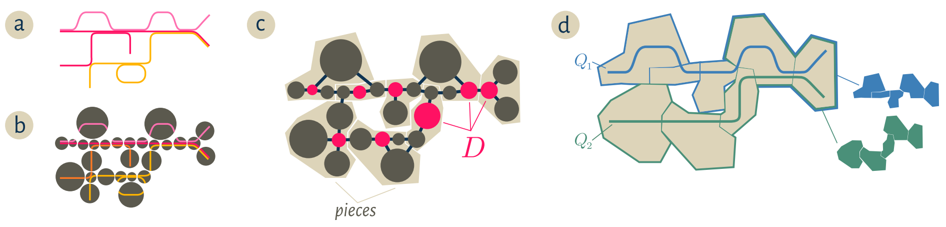

Fig. 1 Starting from a collection of genomic sequences (**a**), we form an assembly graph where nodes represent distinct linear subsequences (**b**). In this assembly graph, known as a *compact De Bruijn graph* [4], nodes may represent many *k*-mers. The original genomic sequences correspond to walks in the graph, and shared nodes between the walks represent shared subsequences. **c** We then identify a subset of nodes *D* called a *dominating set* so that every node in the assembly graph is at distance at most one from some member of *D* (marked pink). We further partition the graph into *pieces* by assigning every node to exactly one of the closest members of *D* (beige regions in **c** and **d**). For a genomic query *Q*, the *neighborhood* of *Q* in this graph is the union of all pieces which share at least one *k*-mer with the query. The colorful subsets of the pieces in **d** correspond to the neighborhoods of the queries Q_1, Q_2

vertex is close to a dominator. The core of the efficient distance computation is based on *distance-truncated transitive fraternal* (*dtf*) *augmentations* [24] which produce a directed graph $\overrightarrow{G}_r$ in which each arc uv is labeled with $\omega(uv)$, the distance from u to v in the original cDBG G.

Importantly, our implementation enhances the algorithm in [24] by computing only $r-1$ rounds of dtf augmentations instead of $2r-1$. Since augmentation is the computationally most expensive part of the pipeline and the running time depends non-linearly on the number of rounds, this vastly improves this algorithm's scalability. To maintain approximation guarantees on the dominating set size with fewer augmentations, we introduce a threshold parameter `domThreshold`(r) which affects the constant factor in our worst-case bound. We selected a threshold (see Additional file 1) that produces r-dominating sets of comparable size to those computed by the algorithm in [24]. Additionally, we found that processing vertices using a minimum in-degree ordering (line 6) was superior to other common orders (e.g., arbitrary, min/max total degree, max in-degree).

After computing an r-dominating set D of G with Algorithm 1, Algorithm 2 partitions the vertices of G into pieces so that each piece $P[v]$ contains a connected set of vertices for which v is a closest member of D (Fig. 1). Finally, we use minimal perfect hashing (`mphfIndex`) [29] to compute an invertible index[1] between pieces and their sequence content in the metagenome.

One feature of this approach is that the dominating set and index only need to be computed once for a given metagenome, independent of the number and content of anticipated queries. Queries can then be performed using Algorithm 3 in time that scales linearly with the size of their *neighborhood*—all genomic content assigned to pieces that contain part of the query.

Our implementations of these algorithms in spacegraphcats can be run on metagenomic data with millions of cDBG nodes (Table 1); indexing takes under an hour, enabling queries to complete in seconds to minutes (Table 2). See Additional file 1, Appendix D for full benchmarking (including cDBG construction). This provides us with a tool to systematically investigate assembly graph neighborhoods.

[1] An invertible function that defines both an index and the corresponding inverted index

Algorithm 1 `rdomset`(G, r)

```
Input: Graph $G$, radius $r$
Output: $r$-dominating set $D$ of $G$
1: $\vec{G}_1 \leftarrow$ orient($G$)
2: for $i \in 2, \ldots, r$ do
3:     $\vec{G}_i \leftarrow$ dtfAugmentation($\vec{G}_{i-1}$)
4: Initialize $d[v] \leftarrow \infty$ and $c[v] \leftarrow 0$ for all $v \in G$
5: $D \leftarrow \emptyset$
6: for all $v \in \vec{G}_r$ sorted by ascending in-degree do
7:     for all $u \in N^-(v)$ do
8:         $d[v] \leftarrow \min(d[v], d[u] + \omega(uv))$
9:     if $d[v] > r$ then
10:        $D \leftarrow D \cup \{v\}$ and $d[v] \leftarrow 0$
11:        for all $u \in N^-(v)$ do
12:            $d[u] \leftarrow \min(d[u], \omega(uv))$
13:            $c[u] \leftarrow c[u] + 1$
14:            if $c[u] >$ domThreshold($r$) then
15:                $D \leftarrow D \cup \{u\}$ and $d[u] \leftarrow 0$
16:                for all $w \in N^-(u)$ do
17:                    $d[w] \leftarrow \min(d[w], \omega(wu))$
18: return $D$
```

Algorithm 2 `indexPieces`$(\mathcal{M}, r)$

```
Input: Metagenome $\mathcal{M}$, radius $r$
Output: Invertible index $I\colon \mathcal{M} \rightarrow \mathcal{P}$; $\mathcal{P}$ is a set of pieces
1: $G \leftarrow$ cDBG($\mathcal{M}$)
2: $D \leftarrow$ rdomset($G, r$)
3: Initialize $\delta[v] \leftarrow v$ for all $v \in D$
4: $U \leftarrow V(G) \setminus D$
5: while $U \neq \emptyset$ do
6:     for $v \in V(G) \setminus U$ do
7:         for $u \in N(v) \cap U$ do
8:             $\delta[u] \leftarrow \delta[v]$
9:             $U \leftarrow U \setminus \{u\}$
10: $\mathcal{P}[v] \leftarrow \{u : \delta[u] = v\}$
11: return mphfIndex($\mathcal{M}, \mathcal{P}$)
```

Algorithm 3 `search`$(I_{\mathcal{M}}, Q)$

```
Input: Index $I_{\mathcal{M}}$, Query $Q$
Output: The query neighborhood $\mathcal{N}_Q$
1: $\mathcal{N}_Q \leftarrow \bigcup_{k \in Q} I_{\mathcal{M}}^{-1}(I_{\mathcal{M}}(k))$
2: return $\mathcal{N}_Q$
```

Table 1 Number of cDBG nodes $|V|$, edge density of cDBG $|E|/|V|$, size of 1-dominating set $|D|$, average query size (k-mers) $\overline{|Q|}$, and average number of pieces in query neighborhood $\overline{|\mathcal{P} \cap \mathcal{N}_Q|}$

Data set	$\|V\|$	$\|E\|/\|V\|$	$\|D\|$	$\overline{\|Q\|}$	$\overline{\|\mathcal{P} \cap \mathcal{N}_Q\|}$
`podarV`	916,041	2.2	542,350	1,475,892	4106
`HuSB1`	13,852,950	2.6	6,724,505	1,112,516	106,091

Queries are the 51 genomes and 23 genome bins fully present in `podarV` and `HuSB1`, respectively

Neighborhood queries enable recovery of relevant unknown genomic content

We first measured how well an inexact query can recover a target genome from a metagenome. For a benchmark data set, we used the `podarV` data set [30]. This is a "mock" metagenome containing genomes from 65 strains and species of bacteria and archaea, each grown independently and rendered into DNA, then combined and sequenced as a metagenome. This metagenome is a commonly used benchmark for assembly [12, 31–33].

To evaluate the effectiveness of neighborhood query at recovering strain variants, we chose three target genomes from `podarV`—*Porphyromonas gingivalis ATCC 33277*, *Treponema denticola ATCC 35405*, and *Bacteroides thetaiotaomicron VPI-5482*—that have many taxonomically close relatives in GenBank. We then used these relatives to query the `podarV` mixture and measure the recovery of the target genome. The results, in Fig. 2a, show that graph neighborhood query can recover 35% or more of some target genomes starting from a relative with Jaccard similarity as low as 1%: even a small number of shared k-mers sufficed to define a much larger neighborhood that contains related genomes.

We next applied neighborhood query to retrieve an unknown genome from `podarV`. Several papers have noted the presence of unexpected sequence in the assemblies of this data, and Awad et al. identify at least two species that differ from the intended mock metagenome contents [12, 31]. One species variant has partial matches to several different *Fusobacterium nucleatum* genomes, while the other incompletely matches to three strains of *Proteiniclasticum ruminis*.

The complete genomes of these two variants are not in public databases and, for the *Proteiniclasticum* variant, cannot be entirely recovered with existing approaches: when we assemble the reads that share k-mers with the available genomes, a marker-based analysis with CheckM estimates that 98.8% of the *Fusobacterium* variant is recovered, while only 72.96% of the *Proteiniclasticum* variant is recovered. We therefore tried using neighborhood queries to expand our knowledge of the *Proteiniclasticum* variant.

Table 2 Time and memory usage of spacegraphcats for Algorithms 1–3 on representative metagenome data

Data set	Algorithm	Time (s)	Memory (MB)
	`rdomset`	78.1	4304
`podarV`	`indexPieces`	359.9	14,108
	`search`	14.9	3463
	`rdomset`	1181.1	60,238
`HuSB1`	`indexPieces`	859.3	40,713
	`search`	67.9	15,228

The times for Algorithm 3 are averaged over all queries (see Table 1). Statistics reported for Algorithm 2 exclude lines 1–2 of pseudocode. Times are rounded to the nearest tenth of a second; memory is rounded to the nearest megabyte

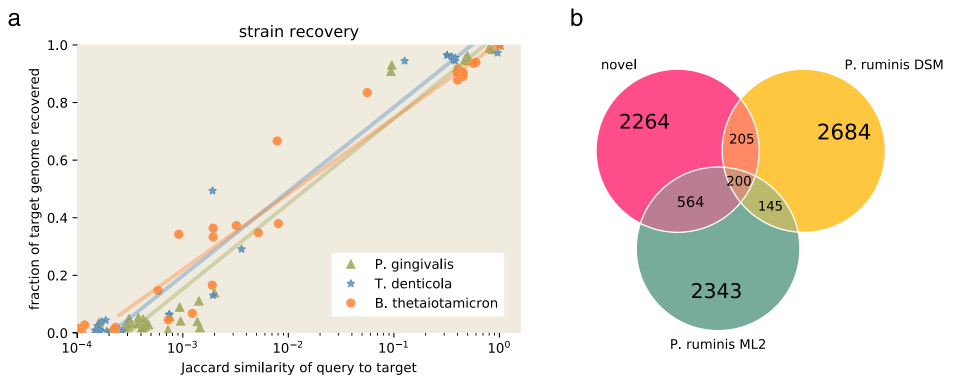

Fig. 2 Neighborhood queries enable recovery of relevant genomic content. **a** Left panel: recovery of each of three target genomes from `podarV` using queries at a variety of Jaccard distances from the target. Recovery is calculated as containment of target genome in query neighborhood. The solid lines represent logarithmic best-fit curves to the points. **b** Right panel: recovery of novel *Proteiniclasticum* content from `podarV`. Nucleotide *k*-mers from two of the three known *P. ruminis* genomes overlapped approximately a megabase of sequence in the query neighborhood, which also contained approximately 2.3 Mbp of unknown sequence; the third known genome, *P. ruminis CGMCC*, was omitted from the figure as it is 99.7% similar to *P. ruminis DSM*. Numbers are in thousands of *k*-mers, estimated via sourmash

We performed a neighborhood query into `podarV` with all three known *Proteiniclasticum* genomes from GenBank. We then extracted the reads overlapping this neighborhood and assembled them with MEGAHIT. The retrieved genome neighborhood for *Proteiniclasticum* contains 2264K novel *k*-mers (Fig. 2b). The reads from the query neighborhood assembled into a 3.1-Mbp genome bin. The assembly is estimated by CheckM to be 100% complete, with 10.3% contamination. The mean amino acid identity between *P. ruminis ML2* and the neighborhood assembly is above 95%, suggesting that this is indeed the genome of the *Proteiniclasticum* variant and that neighborhood query retrieves a full draft genome sequence (see Additional file 1, Appendix G).

Query neighborhoods in a real metagenome do not always assemble well

Real metagenomes may differ substantially from mock metagenomes in size, complexity, and content. In particular, real metagenomes may contain a complex mixture of species and strain variants [34] and the performance of assembly and binning algorithms on these data sets is challenging to evaluate in the absence of ground truth. One recent comparison of single-cell genomes and metagenome-assembled genomes in an ocean environment found that up to 40% of single-cell genome content may be missing in metagenome-assembled genomes [15].

We first ask whether genome query neighborhood sizes in a real metagenome differ from mock metagenomes. We examined genomes inferred from the SB1 sample from the Hu et al. study, in which 6 metagenomic samples taken from various types of oil reservoirs were sequenced, assembled, binned, and computationally analyzed for biochemical function [35]. Examining the 23 binned genomes in GenBank originating from the SB1 sample, we compared the `HuSB1` neighborhood size distribution with the `podarV` data set (Fig. 3a). We saw that more genome bins in `HuSB1` have 1.5× or larger query neighborhoods than do the genomes in `podarV`. This suggests the presence of considerably more local neighborhood content in the real metagenome than in the mock metagenome.

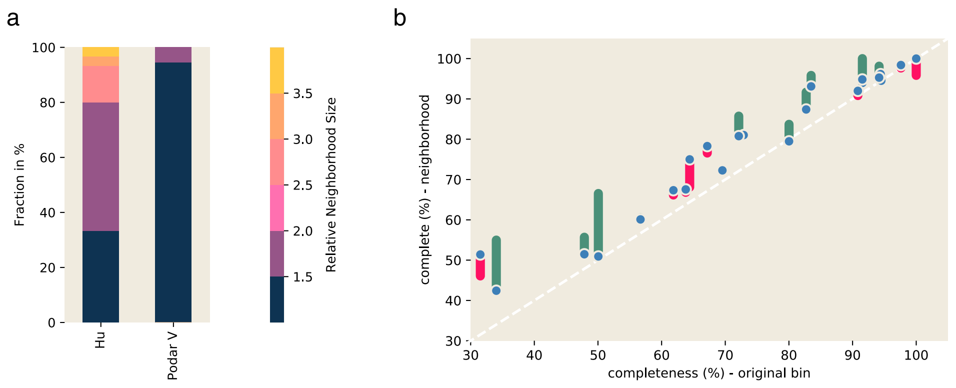

Fig. 3 Query neighborhoods in `HuSB1` metagenome are large and contain additional marker genes. **a** Left panel: neighborhood sizes are larger in `HuSB1` than in `podarV`. Here, we queried `podarV` and `HuSB1` using each of 51 and 23 genomes fully present in the respective data sets and measured the relative size of its neighborhood—a size of 1 indicates that no additional sequence is present in the neighborhood, while a size of 2 indicates that the retrieved neighborhood is twice the size of the query genome. **b** Right panel: query neighborhoods are estimated to be more complete than the original genome bins. We queried `HuSB1` using each of 23 genomes binned from SB1 and assembled the resulting neighborhoods using MEGAHIT and Plass. The blue points represent completeness estimates of MEGAHIT-assembled neighborhoods, while green and pink bars represent the additional or missing content in the Plass assemblies, respectively

We next investigated metagenomic content in the query neighborhoods. As with the unknown variants in `podarV`, we used CheckM to estimate genome bin completeness. The estimated bin completeness for many of the query genomes is low (Additional file 1, Appendix I). To see if the query neighborhoods contain missing marker genes, we assembled reads from the query neighborhoods using MEGAHIT and found this improved the completion metrics (Fig. 3b).

Investigating further, we found that the query neighborhood assemblies contain only between 4 and 56% of the neighborhood *k*-mer content (Additional file 1, Appendix J), suggesting that MEGAHIT is not including many of the reads in the assembly of the query neighborhoods. This could result from high error rates and/or high strain variation in the underlying reads [11, 12].

To attempt the recovery of more gene content from the assemblies, we turned to the Plass amino acid assembler [36]. Plass implements an overlap-based amino acid assembly approach that is considerably more sensitive than nucleotide assemblers and could be more robust to errors and strain variation [37].

When we applied Plass to the reads from the query neighborhoods, we saw a further increase in neighborhood completeness (Fig. 3b). This suggests that the genome bin query neighborhoods contain real genes that are accessible to the Plass amino acid assembler. We note that these are unlikely to be false positives, since CheckM uses a highly specific Hidden Markov Model (HMM)-based approach to detecting marker genes [38].

Some query neighborhoods contain substantial strain variation

If strain variation is contributing to poor nucleotide assembly of marker genes in the query neigborhoods, then Plass should assemble these variants into similar amino acid sequences. Strain variation for unknown genes can be difficult to study due to lack of ground truth, but highly conserved proteins should be readily identifiable.

The *gyrA* gene encodes an essential DNA topoisomerase that participates in DNA supercoiling and was used by [35] as a phylogenetic marker. In the GenBank bins, we found that 15 of the 23 bins contain at least one gyrA sequence (with 18 *gyrA* genes total). We therefore used *gyrA* for an initial analysis of the Plass-assembled neighborhood content for all 23 bins. To avoid confounding effects of random sequencing error in the analysis and increase specificity at the cost of sensitivity, we focused only on high-abundance data: we truncated all reads in the query neighborhoods at any *k*-mer that appears fewer than five times, and ran Plass on these abundance-trimmed reads from each neighborhood. We then searched the gene assemblies with a gyrA-derived HMM, aligned all high-scoring matches, and calculated a pairwise similarity matrix from the resulting alignment.

When we examine all of the high-scoring gyrA protein matches in the hard-trimmed data, we see considerable sequence variation in some query neighborhoods (Fig. 4a). Much of this variation is present in fragmented Plass assemblies; when the underlying nucleotide sequences are retrieved and used to construct a compact De Bruijn graph, the variation is visible as spurs off of a few longer paths (insets in Fig. 4a). When we count the number of well-supported amino acid variants in isolated positions (i.e., ignoring linkage between variants), we see that ten of the 23 neighborhoods have an increased number of gyrA genes, with four neighborhoods gaining a gyrA where none exists in the bin (Additional file 1, Appendix L; see lowest inset in Fig. 4a for one example). Only one neighborhood, *M. bacterium*, loses its gyrA genes due to the stringent *k*-mer abundance trimming. Collectively, the use of the Plass assembler on genome neighborhoods substantially increases the number of gyrA sequences associated with bins.

We see this same pattern for many genes, including *alaS*, *gyrB*, *rpb2* domain 6, *recA*, *rplB*, and *rpsC* (Additional file 1, Appendix M). This shows that multiple variants of those

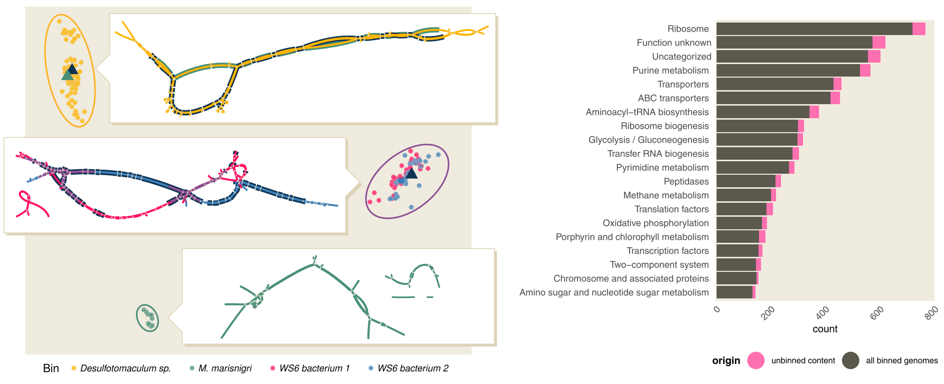

Fig. 4 Query neighborhoods in `HuSB1` contain sequence variants and new genes. **a** Left panel: gyrA has substantial minor sequence variation in several query neighborhoods. In this multidimensional scaling plot, each point represents a distinct gyrA sequence from the Plass assemblies of four representative query neighborhoods, colored by query binned genome. The triangles represent gyrA sequences originating from the query binned genome, if any are present. The inlays are visualizations of assembly graphs of reads that contain *gyrA* sequence in each neighborhood. Unitigs are colored by their cluster of origin; matches to *gyrA* sequences from the bin are highlighted using color from relevant triangle. **b** Right panel: genome neighborhoods re-associate annotated functionality to binned genomes. For each of 23 genome bins originating from `HuSB1`, we found the unbinned content by removing all orthologs found in the binned genomes in [39] and by counting distinct ortholog annotations once. Functional content is distributed throughout pathways present in the binned genomes and increases functionality associated with binned genomes by approximately 13%

proteins are present within at least some of the neighborhoods and implies the presence of underlying nucleotide strain variation. This strain variation may be one reason that nucleotide assembly performs poorly: on average, only 19.6% of Plass-assembled proteins are found within the nucleotide assemblies.

Query neighborhoods assembled with Plass contain additional functional content

In addition to capturing variants close to sequences in the bins, we identify many novel genes in the query neighborhoods. We used KEGG to annotate the Plass-assembled amino acid sequences and then removed any annotations already present for genes in the genome bin. We also ignored homolog abundance such that each homolog is counted only once per neighborhood.

Novel functional content is distributed throughout pathways present in the genome bins and increases functionality associated with binned genomes by approximately 13% (Fig. 4b). This includes orthologs in biologically relevant pathways such as methane metabolism, which are important for biogeochemical cycling in oil reservoirs [35].

Genes in these neighborhoods contain important metabolic functionality expanding the pathways already identified in [35]. We find 40 unique orthologs involved in nitrogen fixation across eight neighborhoods, four of which had no ortholog in the bin. Importantly, we find the ratio of observed orthologs approximately matches that noted in [35], where two thirds of nitrogen fixation functionality is attributable to archaea (29 of 40 orthologs). This is in contrast to most ecological systems where bacteria are the dominant nitrogen fixers [35].

Discussion

Efficient graph algorithms provide novel tools for investigating graph neighborhoods

Recent work has shown that incorporating the structure of the assembly graph into the analysis of metagenome data can provide a more complete picture of gene content [21, 22]. While this has provided evidence that it is useful to analyze sequences that have small graph distance from a query (are in a "neighborhood"), this approach has not been widely adopted. Naïvely, local expansion around many queries in the assembly graph does not scale to these types of analyses due to the overhead associated with searching in a massive graph. The neighborhood index structure described in this work overcomes this computational obstacle and enables rapid exploration of sequence data that is local to a query.

Because a partition into pieces provides an implicit data reduction (the cDBG edge relationships are subsumed by piece membership), the query-independent nature of the index allows many queries to be processed quickly without loading the entire graph into memory. Our approach consequently provides a data exploration framework not otherwise available.

Exploiting the structural sparsity of cDBGs is a crucial component of our algorithms. First, it is necessary to use graph structure to obtain a guarantee that Algorithm 2 finds a small number of pieces since the size of a minimum r-dominating set cannot be approximated better than a factor of $\log n$ in general graphs[2] unless NP $\subseteq$ DTIME$(n^{O(\log\log n)})$ [26]. Without such a guarantee, we cannot be sure that we are achieving significant data

[2]That is, graphs about which we make no structural assumptions.

reduction by grouping cDBG vertices into pieces. Being able to do this in linear time also ensures that indexing and querying can scale to very large data sets. Furthermore, because we utilize a broad structural characterization (bounded expansion) of cDBGs rather than a highly specialized aspect, our methods enable neighborhood-based information retrieval in any domain whose graphs exhibit bounded expansion structure; examples include some infrastructure, social, and communication networks [24, 40, 41].

Neighborhood queries extend genome bins

In both the `podarV` and `HuSB1` metagenomes, neighborhood queries were able to identify additional content likely belonging to query genomes. In the `podarV` mock metagenome, we retrieved a potentially complete genome for an unknown strain based on partial matches to known genomes. In the `HuSB1` metagenome, we increased the estimated completeness of most genome bins—in some cases substantially, e.g., in the case of *P_bacterium 34_609*, we added an estimated 20.9% to the genome bin. In both cases, we rely solely on the structure of the assembly graph to expand the genome bins. We do not make use of sequence composition, contig abundance, or phylogenetic marker genes in our search. Thus graph proximity provides an orthogonal set of information for genome-resolved metagenomics that could be used to improve current binning techniques.

Query neighborhoods from real metagenomes contain substantial strain variation that may block assembly

Previous work suggests that metagenome assembly and binning approaches are fragile to strain variation [11, 12]. This may prevent the characterization of some genomes from metagenomes. The extent of this problem is unknown, although the majority of approaches to genome-resolved metagenomics rely on assembly and thus could be affected.

In this work, we find that some of the sequence missing from genome bins can be retrieved using neighborhood queries. For `HuSB1`, some genome bins are missing as many as 68.5% of marker genes from the original bins, with more than half of the 22 bins missing 20% or more; this accords well with evidence from a recent comparison of single-cell genomes and metagenome-assembled genomes [15], in which it was found that metagenome-assembled genomes were often missing 20 to 40% of single-cell genomic sequence. Neighborhood query followed by amino acid assembly recovers additional content for all but two of the genome bins; this is likely an underestimate, since Plass may also be failing to assemble some content.

When we bioinformatically analyze the function of the expanded genome content from neighborhood queries, our results are consistent with the previous metabolic analyses by [35] and extend the set of available genes by 13%. This suggests that current approaches to genome binning are specific and that the main question is sensitivity, which agrees with a more direct measurement of lost content [15].

Neighborhood queries enable a genome-targeted workflow to recover strain variation

The spacegraphcats analysis workflow described above starts with genome bins. The genome bins are used as a query into the metagenome assembly graph, following which we extract reads from the query neighborhood. We assemble these reads with the Plass amino acid assembler and then analyze the assembly for gene content. We show that the Plass assembly contains strain-level heterogeneity at the amino acid level and that this heterogeneity is supported by underlying nucleotide diversity. Even with stringent error trimming on the underlying reads, we identify at least thirteen novel gyrA sequences in ten genome neighborhoods.

Of note, this workflow explicitly associates the Plass-assembled proteins with specific genome bins, as opposed to a whole-metagenome Plass assembly which recovers protein sequence from the entire metagenome but does not link those proteins to specific genomes. The binning-based workflow connects the increased sensitivity of Plass assembly to the full suite of tools available for genome-resolved metagenome analysis, including phylogenomic and metabolic analysis. However, spacegraphcats does not separate regions of the graph shared in multiple query neighborhoods; existing strain recovery approaches such as DESMAN or MSPminer could be used for this purpose [16, 19].

One future step could be to characterize unbinned genomic content from metagenomes by looking at Plass-assembled marker genes in the metagenome that do not belong to any bin's query neighborhood. This would provide an estimate of the extent of metagenome content remaining unbinned.

Conclusions

The neighborhood query approach described in this work provides an alternative window into metagenome content associated with binned genomes. We extend previous work showing that assembly-based methods are fragile to strain variation, and provide an alternative workflow that substantially broadens our ability to characterize metagenome content. This first investigation focuses on only two data sets, one mock and one real, but the neighborhood indexing approach is broadly applicable to all shotgun metagenomes.

In this initial investigation of neighborhood indexing, we have focused on using neighborhood queries with a genome bin. We recognize that this approach is of limited use in regions where no genome bin is available; spacegraphcats is flexible and performant enough to support alternative approaches such as querying with k-mers belonging to genes of interest.

Potential applications of spacegraphcats in metagenomics include developing metrics for genome binning quality, analyzing pangenome neighborhood structure, exploring r-dominating sets for $r > 1$, extending analyses to colored De Bruijn graphs, and investigating de novo extraction of genomes based on neighborhood content. We could also apply spacegraphcats to analyze the neighborhood structure of assembly graphs overlayed with physical contact information (from, for example, HiC), which could yield new applications in both metagenomics and genomics [42, 43].

More generally, the graph indexing approach developed here may be applicable well beyond metagenomes and sequence analysis. The exploitation of bounded expansion to efficiently compute r-dominating sets on large graphs makes this technique applicable to a broad array of problems.

Materials and methods

Data

We use two data sets: SRR606249 from `podarV` [44] and SRR1976948 (sample SB1) from `hu` [39]. Each data set was first preprocessed to remove low-abundance k-mers as in [45], using `trim-low-abund.py` from khmer v2.1.2 [46] with the parameters `-C 3 -Z 18 -M 20e9 -V -k 31`. We build compact De Bruijn graphs using BCALM v2.2.0 [47]. Stringent read trimming at low-abundance k-mers was done with `trim-low-abund.py` from khmer, with the parameters `-C 5 -M 20e9 -k 31`.

Benchmarking

We measured time and memory usage for Algorithms 1–3 by executing the following targets in the spacegraphcats `conf/Snakefile`: `catlas.csv` for `rdomset`, `contigs.fa.gz.mphf` for `indexPieces`, and `search` for `search`. We report wall time and maximum resident set size, running under Ubuntu 18.04 on an NSF Jetstream virtual machine with 10 cores and 30 GB of RAM [48, 49]. To measure maximum resident set size, we used the memusg script (Jaeho Shin, https://gist.github.com/netj/526585).

Graph denoising

For each data set, we built a compact De Bruijn graph (cDBG) for $k = 31$ by computing the set of unitigs with BCALM [50] and removing all vertices of degree one with a mean k-mer abundance of 1.1 or less. After the removal of these vertices, we then contracted any newly revealed degree-two paths.

Neighborhood indexing and search

We used spacegraphcats to build an r-dominating set for each denoised cDBG and index it. We then performed neighborhood queries with spacegraphcats, which produces a set of cDBG nodes and reads that contributed to them. The full list of query genomes for the *Proteiniclasticum* query is available in Additional file 1, Appendix F, and the NCBI accessions for the *P. gingivalis*, *T. denticola*, and *B. thetaiotamicron* queries are in the directory `pipeline-base` of the paper repository, files `strain-gingivalis.txt`, `strain-denticola.txt`, and `strain-bacteroides.txt`, respectively.

Search results analysis

Query neighborhood size, Jaccard containment, and Jaccard similarity were estimated using modulo hash signatures with a k-mer size of 31 and a scaled factor of 1000, as implemented in sourmash v2.0a9 [51].

Assembly and genome bin analysis

We assembled reads using MEGAHIT v1.1.3 [31] and Plass v2-c7e35 [36], treating the reads as single-ended. Bin completeness was estimated with CheckM 1.0.11, with the `-reduced_tree` argument [38]. Amino acid identity between bins and genomes was calculated using CompareM commit 7cd51276 (https://github.com/dparks1134/CompareM).

Gene targeted analysis

Analysis of specific genes was done with HMMER v3.2.1 [52]. Plass amino acid assemblies were queried with HMMER `hmmscan` using the PFAM domains listed in Additional file 1, Table S7, using a threshold score of 100 [53]. Matching sequences were then extracted from the assemblies for further analysis. To overcome problems associated with comparing non-overlapping sequence fragments, only sequences that overlapped 125 of the most-overlapped 200 residues of the PFAM domain were retained (all sequences shared a minimum overlap of 50 residues with all other sequences). These sequences were aligned with MAFFT v7.407 with the `-auto` argument [54]. Pairwise similarities were calculated using HMMER where the final value represented the number of identical amino acids in the alignment divided by the number of overlapping residues between the sequences. Pairwise distances were visualized using a multidimensional scaling calculated in R using the `cmdscale` function. To visualize the assembly graph structure underlying these amino acid assemblies, we used paladin v1.3.1 to map abundance-trimmed reads back to the Plass amino acid assembly, with `-f 125` to set the minimum ORF length accepted [55]. We extracted the reads that mapped to the gene of interest, created an assembly graph using BCALM v2.2.0 [50], and visualized the graph using Bandage v0.8.1 [56]. We colored nucleotide sequences originating from the bins using the BLAST feature in Bandage.

KEGG analysis

We annotated the Plass assemblies using Kyoto Encyclopedia of Genes GhostKOALA v2.0 [57]. To assign KEGG ortholog function, we used methods implemented at https://github.com/edgraham/GhostKoalaParser release 1.1.

Supplementary information

Supplementary information accompanies this paper at https://doi.org/10.1186/s13059-020-02066-4.

Additional file 1: Contains supplementary text, figures, and tables.
Additional file 2: Contains the review history.

Peer review information

Kevin Pang was the primary editor of this article and managed its editorial process and peer review in collaboration with the rest of the editorial team.

Acknowledgements
This work arose from the Barnraising for Data-Intensive Discovery at Mt. Desert Island Biological Lab in May 2016. We thank Erich Schwarz, Martin Steinegger, Johannes Söding, Mark Blaxter, and members of the Data Intensive Biology lab at UC Davis for discussion and feedback.

Review history

The review history is available as Additional file 2.

Authors' contributions

DM, MPO, FR, and BDS designed and implemented the algorithms; CTB, DM, MPO, and FR developed the software; CTB and TR conducted the biological data analysis; CTB and BDS supervised the work. All authors interpreted the results, wrote the text, created the figures, and approved the submitted paper.

Authors' information

Twitter handles: @ctitusbrown (C. Titus Brown); @domoritz (Dominik Moritz); @quantumgravitas (Felix Reidl); @ReiterTaylor (Taylor Reiter); @BlairDSullivan (Blair D. Sullivan).

Funding
This work is funded in part by the Gordon and Betty Moore Foundation's Data-Driven Discovery Initiative through Grants GBMF4551 to C. Titus Brown, GBMF4553 to Jeffrey Heer, and GBMF4560 to Blair D. Sullivan.

Availability of data and materials

The data sets used in this paper are available in the Sequence Read Archive under accessions SRR606249 (`podarV`) and SRR1976948 (`hu`, sample SB1).
The source code for the index construction and search is available at https://github.com/spacegraphcats/spacegraphcats [23]. It is implemented in Python 3 under the 3-Clause BSD License. Version 1.1, used in this paper, is archived at DOI: 10.5281/zenodo.2505206.
Snakemake [58] workflows to reproduce all of the analysis are available at https://github.com/spacegraphcats/2018-paper-spacegraphcats/, and Jupyter Notebooks to recreate Figs. 1, 2, 3, and 4b are in that same repository [59]. The

notebooks rely on the numpy, matplotlib, pandas, scipy, and Vega-Lite libraries [60–64]. The workflow repository is archived at DOI: https://doi.org/10.5281/zenodo.2592780.

Ethics approval and consent to participate
Not applicable.

Competing interests
The authors declare that they have no competing interests.

Author details
[1]Department of Population Health and Reproduction, University of California Davis, Davis, USA. [2]Paul G. Allen School of Computer Science and Engineering, University of Washington, Seattle, USA. [3]Department of Computer Science, NC State University, Raleigh, USA.

Received: 14 June 2019 Accepted: 29 May 2020
Published online: 06 July 2020

References

1. Quince C, Walker AW, Simpson JT, Loman NJ, Segata N. Shotgun metagenomics, from sampling to analysis. Nat Biotechnol. 2017a;35(9):833–44. https://doi.org/10.1038/nbt.3935.
2. Pell J, et al. Scaling metagenome sequence assembly with probabilistic De Bruijn graphs. PNAS. 2012;109(33): 13272–7. https://doi.org/10.1073/pnas.1121464109.
3. Laczny CC, Kiefer C, Galata V, Fehlmann T, Backes C, Keller A. Busybee web: metagenomic data analysis by bootstrapped supervised binning and annotation. Nucleic Acids Res. 2017gkx348. https://doi.org/10.1093/nar/gkx348.
4. Lin H, Liao Y. Accurate binning of metagenomic contigs via automated clustering sequences using information of genomic signatures and marker genes. Sci Rep. 2016;6:24175.
5. Parks DH, Rinke C, Chuvochina M, Chaumeil P-A, Woodcroft BJ, Evans PN, Hugenholtz P, Tyson GW. Recovery of nearly 8,000 metagenome-assembled genomes substantially expands the tree of life. Nat Microbiol. 2017;2(11): 1533–42. https://doi.org/10.1038/s41564-017-0012-7.
6. Tully BJ, Graham ED, Heidelberg JF. The reconstruction of 2,631 draft metagenome-assembled genomes from the global oceans. Sci Data. 2018;5:170203. https://doi.org/10.1038/sdata.2017.203.
7. Stewart RD, Auffret MD, Warr A, Wiser AH, Press MO, Langford KW, Liachko I, Snelling TJ, Dewhurst RJ, Walker AW, Roehe R, Watson M. Assembly of 913 microbial genomes from metagenomic sequencing of the cow rumen. Nat Commun. 2018;9(1). https://doi.org/10.1038/s41467-018-03317-6.
8. Delmont TO, Quince C Shaiber, Esen ÖC, Lee STM, Rappé MS, McLellan SL, Lücker S, Eren AM. Nitrogen-fixing populations of planctomycetes and proteobacteria are abundant in surface ocean metagenomes. Nat Microbiol. 2018;3(7):804–13. https://doi.org/10.1038/s41564-018-0176-9.
9. Hug LA, Baker BJ, Anantharaman K, Brown CT, Probst AJ, Castelle CJ, Butterfield CN, Hernsdorf AW, Amano Y, Ise K, Suzuki Y, Dudek N, Relman DA, Finstad KM, Amundson R, Thomas BC, Banfield JF. A new view of the tree of life. Nat Microbiol. 2016;1(5). https://doi.org/10.1038/nmicrobiol.2016.48.
10. Pasolli E, Asnicar F, Manara S, Zolfo M, Karcher N, Armanini F, Beghini F, Manghi P, Tett A, Ghensi P, Collado MC, Rice BL, DuLong C, Morgan XC, Golden CD, Quince C, Huttenhower C, Segata N. Extensive unexplored human microbiome diversity revealed by over 150, 000 genomes from metagenomes spanning age, geography, and lifestyle. Cell. 2019;176(3):649–62.e20. https://doi.org/10.1016/j.cell.2019.01.001.
11. Sczyrba A, Hofmann P, Belmann P, Koslicki D, Janssen S, Dröge J, Gregor I, Majda S, Fiedler J, Dahms E, Bremges A, Fritz A, Garrido-Oter R, Jørgensen TS, Shapiro N, Blood PD, Gurevich A, Bai Y, Turaev D, DeMaere MZ, Chikhi R, Nagarajan N, Quince C, Meyer F, Balvočiūtė M, Hansen LH, Sørensen SJ, Chia BKH, Denis B, Froula JL, Wang Z, Egan R, Kang DD, Cook JJ, Deltel C, Beckstette M, Lemaitre C, Peterlongo P, Rizk G, Lavenier D, Wu Y-W, Singer SW, Jain C, Strous M, Klingenberg H, Meinicke P, Barton MD, Lingner T, Lin H-H, Liao Y-C, Silva GGZ, Cuevas DA, Edwards RA, Saha S, Piro VC, Renard BY, Pop M, Klenk H-P, Göker M, Kyrpides NC, Woyke T, Vorholt JA, Schulze-Lefert P, Rubin EM, Darling AE, Rattei T, McHardy AC. Critical assessment of metagenome interpretation—a benchmark of metagenomics software. Nat Methods. 2017;14(11):1063–71. https://doi.org/10.1038/nmeth.4458.
12. Awad S, Irber L, Brown CT. Evaluating metagenome assembly on a simple defined community with many strain variants. 2017. https://www.biorxiv.org/content/early/2017/07/03/155358.
13. Brown CT. Strain recovery from metagenomes. Nat Biotechnol. 2015;33(10):1041–3. https://doi.org/10.1038/nbt.3375.
14. Brito IL, Alm EJ. Tracking strains in the microbiome: insights from metagenomics and models. Front Microbiol. 2016;7. https://doi.org/10.3389/fmicb.2016.00712.
15. Alneberg J, Karlsson CMG, Divne A-M, Bergin C, Homa F, Lindh MV, Hugerth LW, Ettema TJG, Bertilsson S, Andersson AF, Pinhassi J. Genomes from uncultivated prokaryotes: a comparison of metagenome-assembled and single-amplified genomes. Microbiome. 2018;6(1). https://doi.org/10.1186/s40168-018-0550-0.
16. Quince C, Delmont TO, Raguideau S, Alneberg J, Darling AE, Collins G, Eren AM. DESMAN: a new tool for de novo extraction of strains from metagenomes. Genome Biol. 2017b;18(1). https://doi.org/10.1186/s13059-017-1309-9.
17. Nayfach S, Rodriguez-Mueller B, Garud N, Pollard KS. An integrated metagenomics pipeline for strain profiling reveals novel patterns of bacterial transmission and biogeography. Genome Res. 2016;26(11):1612–25. https://doi.org/10.1101/gr.201863.115.
18. Garrison E. Graphical pangenomics. PhD thesis: Cambridge University; 2018. https://doi.org/10.5281/zenodo.1463032. As submitted, awaiting viva (defense) and further revision.
19. Onate FP, Chatelier EL, Almeida M, Cervino ACL, Gauthier F, Magoules F, Ehrlich SD, Pichaud M. MSPminer: abundance-based reconstitution of microbial pan-genomes from shotgun metagenomic data. Bioinformatics. 2018. https://doi.org/10.1093/bioinformatics/bty830.

20. Petersen JM, Kemper A, Gruber-Vodicka H, Cardini U, van der Geest M, Kleiner M, Bulgheresi S, Mußmann M, Herbold C, Seah BKB, Antony CP, Liu D, Belitz A, Weber M. Chemosynthetic symbionts of marine invertebrate animals are capable of nitrogen fixation. Nat Microbiol. 2016;2(1). https://doi.org/10.1038/nmicrobiol.2016.195.
21. Olekhnovich EI, Vasilyev AT, Ulyantsev VI, Kostryukova ES, Tyakht AV. MetaCherchant: analyzing genomic context of antibiotic resistance genes in gut microbiota. Bioinformatics. 2017;34(3):434–44. https://doi.org/10.1093/bioinformatics/btx681.
22. Barnum TP, Figueroa IA, Carlström CI, Lucas LN, Engelbrektson AL, Coates JD. Genome-resolved metagenomics identifies genetic mobility, metabolic interactions, and unexpected diversity in perchlorate-reducing communities. ISME J. 2018;12(6):1568–81. https://doi.org/10.1038/s41396-018-0081-5.
23. Brown CT, Moritz D, O'Brien MP, Reidl F, Sullivan BD. spacegraphcats, v1.0. 2018. http://dx.doi.org/10.5281/zenodo.1478025.
24. Reidl F. Structural sparseness and complex networks. 2016. http://publications.rwth-aachen.de/record/565064. Aachen, Techn. Hochsch., Diss., 2015.
25. Karp RM. Reducibility among combinatorial problems. In: Complexity of computer computations. Springer; 1972. p. 85–103. https://doi.org/10.1007/978-1-4684-2001-2_9.
26. Chlebík, M, Chlebíková J. Approximation hardness of dominating set problems in bounded degree graphs. Inf Comput. 2008;206(11):1264–75.
27. Downey RG, Fellows MR. Parameterized complexity: Springer Science & Business Media; 2012.
28. de Mendez PO, et al. Sparsity: graphs, structures, and algorithms, volume 28. 2012. https://doi.org/10.1007/978-3-642-27875-4.
29. Limasset A, Rizk G, Chikhi R, Peterlongo P. Fast and scalable minimal perfect hashing for massive key sets. CoRR. 017;abs/1702.03154. http://arxiv.org/abs/1702.03154.
30. Shakya M, Quince C, Campbell JH, Yang ZK, Schadt CW, Podar M. Comparative metagenomic and rrna microbial diversity characterization using archaeal and bacterial synthetic communities. Environ Microbiol. 2013a;15(6): 1882–99. ISSN 1462-2920. https://doi.org/10.1111/1462-2920.12086.
31. Li D, Luo R, Liu C-M, Leung C-M, Ting HF, Sadakane K, Yamashita H, Lam T-W. MEGAHIT v1.0: a fast and scalable metagenome assembler driven by a dvanced methodologies and community practices. Methods. 2016;102:3–11. https://doi.org/10.1016/j.ymeth.2016.02.020.
32. Seah BKB, Gruber-Vodicka HR. gbtools: interactive visualization of metagenome bins in r. Front Microbiol. 2015;6. https://doi.org/10.3389/fmicb.2015.01451.
33. Nurk S, Meleshko D, Korobeynikov A, Pevzner PA. metaspades: a new versatile metagenomic assembler. Genome Res. 2017;27(5):824–34.
34. Sharon I, Kertesz M, Hug LA, Pushkarev D, Blauwkamp TA, Castelle CJ, Amirebrahimi M, Thomas BC, Burstein D, Tringe SG, Williams KH, Banfield JF. Accurate, multi-kb reads resolve complex populations and detect rare microorganisms. Genome Res. 2015;25(4):534–43. https://doi.org/10.1101/gr.183012.114.
35. Hu P, Tom L, Singh A, Thomas BC, Baker BJ, Piceno YM, Andersen GL, Banfield JF. Genome-resolved metagenomic analysis reveals roles for candidate phyla and other microbial community members in biogeochemical transformations in oil reservoirs. mBio. 2016a;7(1). https://doi.org/10.1128/mbio.01669-15.
36. Steinegger M, Mirdita M, Soding J. Protein-level assembly increases protein sequence recovery from metagenomic samples manyfold. 2018. https://doi.org/10.1101/386110.
37. Yang Y, Yooseph S. SPA: a short peptide assembler for metagenomic data. Nucleic Acids Res. 2013;41(8):e91. https://doi.org/10.1093/nar/gkt118.
38. Parks DH, Imelfort M, Skennerton CT, Hugenholtz P, Tyson GW. CheckM: assessing the quality of microbial genomes recovered from isolates, single cells, and metagenomes. Genome Res. 2015;25(7):1043–55. https://doi.org/10.1101/gr.186072.114.
39. Hu P, Tom L, Singh A, Thomas BC, Baker BJ, Piceno YM, Andersen GL, Banfield JF. Genome-resolved metagenomic analysis reveals roles for candidate phyla and other microbial community members in biogeochemical transformations in oil reservoirs. MBio. 2016b;7(1):01669–15. https://doi.org/10.1128/mBio.01669-15.
40. Demaine ED, Reidl F, Rossmanith P, Villaamil FS, Sik-dar S, Sullivan BD. Structural sparsity of complex networks: Bounded expansion in random models and realworld graphs. J Comput Syst Sci. 2019;105:199–241. https://doi.org/10.1016/j.jcss.2019.05.004.
41. Nadara W, Pilipczuk M, Rabinovich R, Reidl F, Siebertz S. Empirical evaluation of approximation algorithms for generalized graph coloring and uniform quasi-wideness. In: D'Angelo G, editor. 17th International Symposium on Experimental Algorithms, SEA 2018, June 27-29 2018, L'Aquila, Italy, volume 103 of LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik; 2018. p. 14:1–16. https://doi.org/10.4230/LIPIcs.SEA.2018.14.
42. Marbouty M, Cournac A, Flot J-F, Marie-Nelly H, Mozziconacci J, Koszul R. Metagenomic chromosome conformation capture (meta3c) unveils the div ersity of chromosome organization in microorganisms. eLife. 2014;3. https://doi.org/10.7554/elife.03318.
43. Beitel CW, Froenicke L, Lang JM, Korf IF, Michelmore RW, Eisen JA, Darling AE. Strain- and plasmid-level deconvolution of a synthetic metagenome by sequencing proximity ligation products. PeerJ. 2014;2:e415. https://doi.org/10.7717/peerj.415.
44. Shakya M, et al. Comparative metagenomic and rrna microbial diversity characterization using archaeal and bacterial synthetic communities. Environ Microbiol. 2013b;15(6):1882–99. https://doi.org/10.1111/1462-2920.12086.
45. Zhang Q, Awad S, Brown CT. Crossing the streams: a framework for streaming analysis of short DNA sequencing reads. 2015. https://doi.org/10.7287/peerj.preprints.890v1.
46. Standage D, yari A, Cohen LJ, Crusoe MR, Head T, Irber L, Joslin SEK, Kingsley NB, Murray KD, Neches R, Scott C, Shean R, Steinbiss S, Sydney C, Brown CT. khmer release v2.1: software for biological sequence analysis. J Open Source Softw. 2017;2(15):272. https://doi.org/10.21105/joss.00272.
47. Chikhi R, Limasset A, Medvedev P. Compacting de bruijn graphs from sequencing data quickly and in low memory. Bioinformatics. 2016a;32(12):i201–8. https://doi.org/10.1093/bioinformatics/btw279.

48. Stewart CA, Turner G, Vaughn M, Gaffney NI, Cockerill TM, Foster I, Hancock D, Merchant N, Skidmore E, Stanzione D, Taylor J, Tuecke S. Jetstream. In: Proceedings of the 2015 XSEDE Conference on Scientific Advancements Enabled by Enhanced Cyberinfrastructure - XSEDE'15. ACM Press; 2015. https://doi.org/10.1145/2792745.2792774.
49. Towns J, Cockerill T, Dahan M, Foster I, Gaither K, Grimshaw A, Hazlewood V, Lathrop S, Lifka D, Peterson GD, Roskies R, Scott JR, Wilkens-Diehr N. XSEDE: accelerating scientific discovery. Comput Sci Eng. 2014;16(5):62–74. https://doi.org/10.1109/mcse.2014.80.
50. Chikhi R, Limasset A, Medvedev P. Compacting De Bruijn graphs from sequencing data quickly and in low memory. Bioinformatics. 2016b;32(12):i201–8.
51. Brown CT, Irber L, Cohen L. dib-lab/sourmash: v1.0. 2016. https://doi.org/10.5281/zenodo.153989.
52. Eddy SR, HMMER Development Team. Hmmer v3.2.1. 2018. http://hmmer.org/. Accessed 8 May 2020.
53. Finn RD, Coggill P, Eberhardt RY, Eddy SR, Mistry J, Mitchell AL, Potter SC, Punta M, Qureshi M, Sangrador-Vegas A, Salazar GA, Tate J, Bateman A. The pfam protein families database: towards a more sustainable future. Nucleic Acids Res. 2015;44(D1):D279–85. https://doi.org/10.1093/nar/gkv1344.
54. Katoh K, Standley DM. MAFFT multiple sequence alignment software version 7: Improvements in performance and usability. Mol Biol Evol. 2013;30(4):772–80. https://doi.org/10.1093/molbev/mst010.
55. Westbrook A, Ramsdell J, Schuelke T, Normington L, Bergeron RD, Thomas WK, MacManes MD. PALADIN: protein alignment for functional profiling whole metagenome shotgun data. Bioinformatics. 2017;33(10):1473–8. https://doi.org/10.1093/bioinformatics/btx021.
56. Wick RR, Schultz MB, Zobel J, Holt KE. Bandage: interactive visualization ofde novogenome assemblies: Fig. 1. Bioinformatics. 2015;31(20):3350–2. https://doi.org/10.1093/bioinformatics/btv383.
57. Kanehisa M, Sato Y, Morishima K. BlastKOALA and GhostKOALA: KEGG tools for functional characterization of genome and metagenome sequences. J Mol Biol. 2016;428(4):726–31. https://doi.org/10.1016/j.jmb.2015.11.006.
58. Koster J, Rahmann S. Snakemake–a scalable bioinformatics workflow engine. Bioinformatics. 2012;28(19):2520–22. https://doi.org/10.1093/bioinformatics/bts480.
59. Kluyver T, Ragan-Kelley B, Pérez F, Granger BE, Bussonnier M, Frederic J, Kelley K, Hamrick JB, Grout J, Corlay S, et al. Jupyter notebooks-a publishing format for reproducible computational workflows. In: ELPUB. New York: IEEE; 2016. p. 87–90.
60. van der Walt S, Varoquaux G. The NumPy array: a structure for efficient numerical computation. Comput Sci Eng. 2011;13(2):22–30. https://doi.org/10.1109/mcse.2011.37.
61. Hunter JD. Matplotlib: a 2d graphics environment. Comput Sci Eng. 2007;9(3):90–5. https://doi.org/10.1109/mcse.2007.55.
62. McKinney W. pandas: a foundational python library for data analysis and statistics. Python High Perform Sci Comput. 20111–9.
63. Jones E, Oliphant T, Peterson P, et al. SciPy: open source scientific tools for Python. 2001. http://www.scipy.org/. Accessed 8 May 2020.
64. Satyanarayan A, Moritz D, Wongsuphasawat K, Heer J. Vega-lite: a grammar of interactive graphics. IEEE Trans Vis Comput Graph. 2017;23(1):341–50. https://doi.org/10.1109/tvcg.2016.2599030.

Publisher's Note