# MetaGraph: Indexing and Analysing Nucleotide Archives at Petabase-scale

Mikhail Karasikov, [1,2,3,4,‡] Harun Mustafa, [1,2,3,4,‡] Daniel Danciu, [1,2] Christopher Barber, [1,2] Marc Zimmermann, [1,2] Gunnar Rätsch, [1,2,3,4,*] and André Kahles [1,2,3,4,*]

[1]Biomedical Informatics Group, Department of Computer Science, ETH Zurich, Zurich, Switzerland
[2]Biomedical Informatics Research, University Hospital Zurich, Zurich, Switzerland
[3]Swiss Institute of Bioinformatics, Zurich, Switzerland
[4]Department of Biology, ETH Zurich, Zurich, Switzerland

([‡]Equal contribution. [*]To whom correspondence should be addressed.)

{raetsch,andre.kahles}@inf.ethz.ch

**Abstract**

The amount of biological sequencing data available in public repositories is growing exponentially, forming an invaluable biomedical research resource. Yet, making all this sequencing data searchable and easily accessible to life science and data science researchers is an unsolved problem. We present *MetaGraph*, a versatile framework for the scalable analysis of extensive sequence repositories. *MetaGraph* efficiently indexes vast collections of sequences to enable fast search and comprehensive analysis. A wide range of underlying data structures offer different practically relevant trade-offs between the space taken by an index and its query performance. Achieving compression ratios of up to 1,000-fold over the already compressed raw input data, *MetaGraph* indexes can represent the content of large sequencing archives in the working memory of a single compute server. We demonstrate our framework's scalability by indexing over 1.4 million whole genome sequencing (WGS) records from NCBI's Sequence Read Archive, representing a total input of more than three petabases. *MetaGraph*provides a flexible methodological framework allowing for index construction to be scaled from consumer laptops to distribution onto a cloud compute cluster for processing terabases to petabases of input data. Notably, processing of data sets ranging from 1 TB of raw WGS reads to 20 TB of human RNA-sequencing data results in indexes whose memory footprints are small enough to host on standard desktop workstations.

Besides demonstrating the utility of *MetaGraph* indexes on key applications, such as experiment discovery, sequence alignment, error correction, and differential assembly, we make a wide range of indexes available as a community resource, including indexes of over 450,000 microbial WGS records, more than 110,000 fungi WGS records, and more than 40,000 whole metagenome sequencing records. A subset of these indexes is made available online for interactive queries. All indexes will be available for download and in the cloud. In total, indexes comprising more than 1 million sequencing records are available for download.

As an example of our indexes' integrative analysis capabilities, we introduce the concept of differential assembly, which allows for the extraction of sequences present in a foreground set of samples but absent in a given background set. We apply this technique to differentially assemble contigs to identify pathogenic agents transfected via human kidney transplants. In a second example, we indexed more than 20,000 human RNA-Seq records from the TCGA and GTEx cohorts and use them to extract transcriptome features that are hard to characterize using a classical linear reference. We discovered over 200 trans-splicing events in GTEx and found broad evidence for tissue-specific non-A-to-I RNA-editing in GTEx and TCGA.

# 1 Introduction

For more than a decade, continuing innovation in the area of high-throughput sequencing has propelled research in the biomedical domain and led to an exponential growth in worldwide sequencing capacity [50]. As a consequence, sequencing costs for entire human genomes have dropped well below the critical mark of 1,000 USD per sample and the new critical line of 100 USD will likely be reached in the near future. This also results in an exponential growth of sequencing data available in public and controlled-access repositories. The number of sequenced nucleotides contained in the European Nucleotide Archive (ENA) currently doubles every 27 months, resulting in a current size of close to $1.6 \cdot 10^{16}$ nucleotide bases (16 petabases) [8] of raw read data. Transmitting the entirety of such a data set across a wire for any kind of access is clearly uneconomical and renders it currently virtually inaccessible to the broader research community for comprehensive analyses. However, even for defined subsets of samples that are collected within larger-scale studies by international consortia, such as The Cancer Genome Atlas (TCGA) [52], the Genotype Tissue Expression (GTEx) [34] project, or the MetaSUB project [15], the entire data of a single study can comprise hundreds of terabytes, making access complicated.

The classical pattern for accessing sequencing data on public repositories is to identify relevant samples using descriptive metadata and to extract a copy or a slice of the data for further processing. More recently, repositories started mirroring their contents into cloud storage, addressing the download problem, but at the same time often creating additional costs for off-the-premise compute. Today's existing infrastructure is largely adapted to this pattern of access, only indexing the metadata (e.g., sample and study ID, organism name, taxonomic information, related publications, etc.) of all samples to make it accessible and easily searchable. However, any query involving the raw sequencing data itself requires a copy of the data, complicating its analysis and reducing its accessibility for many researchers. To address this issue, we propose a scalable approach to index such large repositories of sequencing data, transforming them into a highly compressed and more accessible representation for downstream analysis. A main focus of this work is thereby on scalability, allowing the processing of sequence collections on a petabase scale.

Driven by the open-data movement and the advances in high-throughput sequencing technology, it is our expectation that the number of studies aggregating large cohorts of samples will further increase in the near future. In fact, we envision that making such data shareable and searchable will become a problem of high practical relevance.

As indexing large sequence collections also poses interesting algorithmic questions, different technical solutions addressing these questions have already been proposed in the recent past. Naturally, a first focus lies on making the genetic variation in large cohorts, especially in human, accessible for biomedical research and medicine. Only very recently frameworks for variation graphs, such as VG [19], and methods for compressing haplotypes [17] or paths in graphs more generally [38], have improved variation-aware alignment and variant calling in general [24]. While successful for the analysis of single-species cohorts, these methods struggle to represent the large variability present in distantly related organisms or in metagenomic applications.

Hence, a second focus of the algorithmic work has been put on the *sequence* or *experiment discovery problem*: querying a sequence of interest (e.g., a transcript or an entire genome) against all samples available in sequence repositories. For collections of assembled genomes, the BLAST approach [6] has been in heavy use since 30 years, but lacks the scalability to allow for high throughput searches on highly diverse sequence collections or to allow for the search in raw, unassembled reads.

The methods currently available for solving the experiment discovery problem can be grouped into three main categories: i) Methods based on sketching techniques, which summarize the input data using one or multiple hashing operations and then use these summaries (sketches) to estimate distances between query and target. Examples are MASH [42, 41] and KrakenUniq [13].

ii) Methods employing Bloom filter based data structures to allow for approximate membership queries. Examples are (Split)-Sequence Bloom Tree [48, 49, 23], Bloom Filter Trie [25], BIGSI [12], and COBS [11]. iii) Methods for the exact representation of *annotated de Bruijn graphs*, also called *colored de Bruijn graph*, storing additional metadata as labels of its nodes or edges [27]. Examples are Mantis [44, 4], VARI [37], and others [2, 32]. A major challenge faced by all existing methods is to unite the ability to efficiently operate on petabase scale input data with the capacity for fast and versatile query operations. Further details on all the above methods together with their specific limitations are provided in Supplemental Section B.

In addition to the more recent use in addressing the experiment discovery problem, de Bruijn graphs are also known as a classic framework for read set representation in sequence assembly in both single-sample [10, 39, 20, 21, 32] and multi-sample [27, 51] settings. Of these, the HipMER [20] and MetaHipMER [21] single-sample assemblers scale to massive data sets stored in distributed hash tables. None of these methods, however, tackles the problem of assembly from an integrative analysis perspective. Examples of such queries can include a *core genome assembly*, where sequences common to all samples in a cohort are desired [36, 35], or a *differential assembly*, where sequences shared by a foreground cohort and absent from a background cohort are desired. We introduce the latter concept as an application made tractable at scale by the framework proposed in this work.

To bridge this evident gap in the landscape of sequence analysis tools for large data collections, we present *MetaGraph*, a versatile framework for indexing and analysis of biological sequence libraries at petabase scale. While the approach is not restricted to any specific input alphabet, for the remainder of this work, we will focus on nucleotide sequences originating from DNA or RNA sequencing samples. *MetaGraph* enables building indexes of large collections of sequences employing an annotated *de Bruijn* graph for sequence search and assembly.

The *MetaGraph* framework provides a wide range of compressed data structures for transforming very large sequencing archives into $k$-mer dictionaries, associating each $k$-mer with labels representing metadata associated with its originating sequences.

The data structures underlying *MetaGraph* are designed to balance the trade-off between the space taken by the index and the time needed for query operations. A main design goal of our framework is to allow both performing experiments on single desktop computers and scaling up to distributed compute clusters. This is achieved through a modular approach, efficient parallelization and the computation in external memory. We describe these aspects in the first part of the Results section along with our assessment of scalability.

We then outline results using *MetaGraph* to index data from a diverse collection of public sources, ranging from large RNA-Seq cohorts like TCGA [52] and GTEx [34], over vast archives of whole genome sequencing (WGS) samples comprising over 1 million samples of microbial, fungal, plant, and metazoa organisms currently available in the Sequence Read Archive (SRA) [30], to large sets of highly diverse whole metagenome sequencing samples, like the MetaSUB [15] set or all available human gut metagenome samples. The total amount of sequences indexed in these graphs exceeds by far the crucial figure of one petabase and at last makes this data fully and efficiently searchable by sequence.

Finally, we use these and some smaller data sets not only to demonstrate the scalability and performance of *MetaGraph* but also to demonstrate how the graph indexes can be used for biological discovery.

# 2 Results

## 2.1 A powerful framework for efficient sequence representation

The *MetaGraph index* consists of two main components: i) a de Bruijn graph and ii) its annotating metadata (**Figure 1**, middle right). The graph is represented by a *k-mer dictionary*

mapping all $k$-mers observed in the input sequences to positive integer identifiers assigned to them. These $k$-mers serve as elementary tokens in all operations on the *MetaGraph* index.

The metadata on the graph is a *binary relation* between $k$-mers and their labels representing any categorical features, such as source sample IDs and quantized expression levels. This relation is represented as a sparse binary matrix called the *annotation matrix*, with one row for each $k$-mer and one column for each label. The $(i, j)^{\text{th}}$ element of this matrix has value 1 if and only if the $i^{\text{th}}$ $k$-mer is associated with the $j^{\text{th}}$ label. This matrix can be of an enormous size, containing up to 1 trillion rows and 1 million columns. Yet, due to its sparsity, it can be efficiently compressed.

A notable feature of our framework is that its theoretical concepts, as well as their actual implementation, support the indexing of sequences over arbitrary finite alphabets. Thus, *MetaGraph* can be used for indexing biological sequences of all kinds, such as raw DNA/RNA sequencing reads, assembled genomes, but also protein sequences.

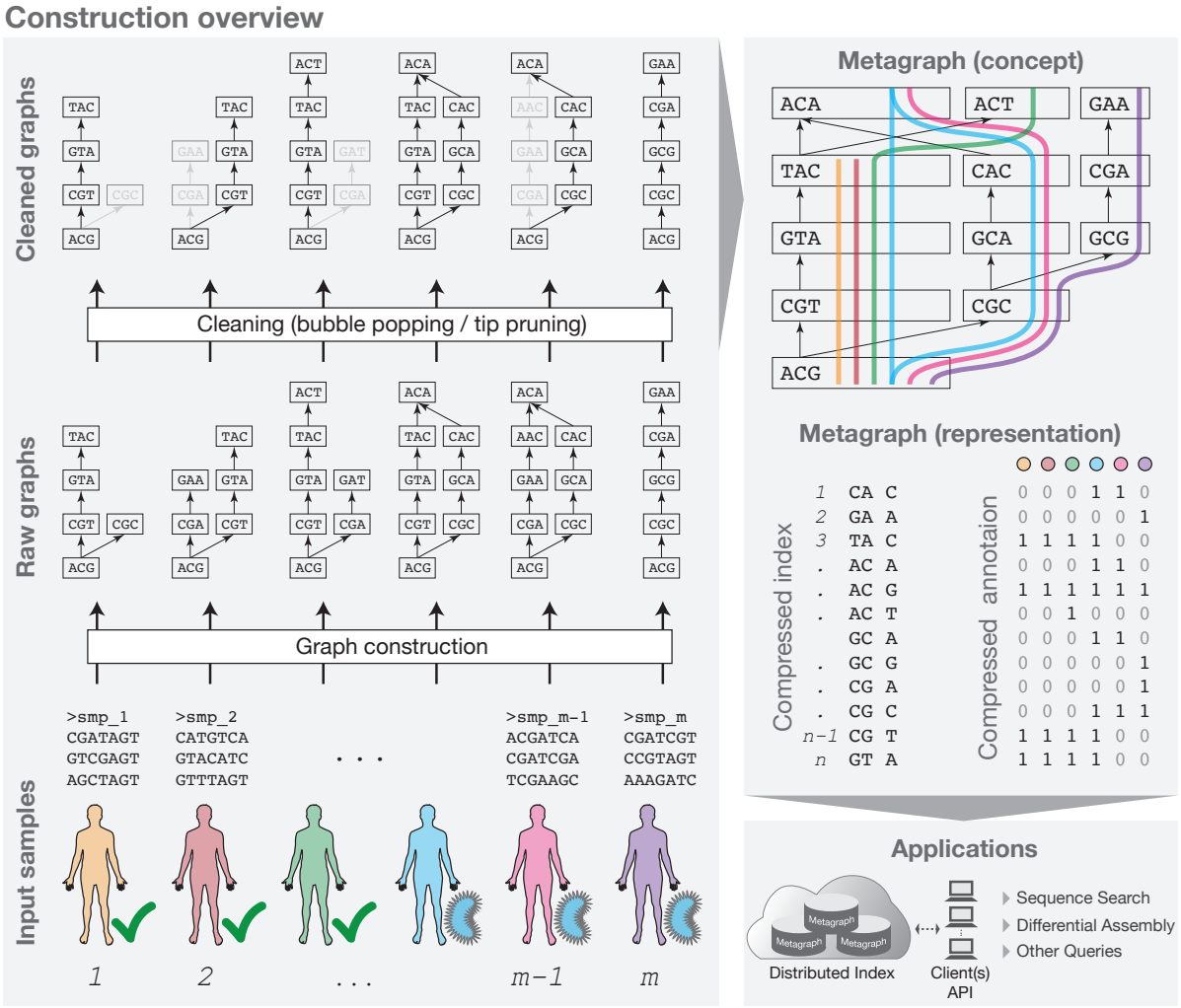


**Figure 1: The *MetaGraph* framework** – Schematic overview of graph construction and representation. **Left:** Individual sequencing samples are assembled into raw graphs which are then cleaned to remove noisy and erroneous paths. **Right:** Individual graphs are combined into the *MetaGraph* index (top), consisting of the compressed sequence index and a compressed annotation matrix (middle). *MetaGraph* is then used as the basis for downstream applications, such as sequence search, differential assembly, and other queries (bottom).

The framework is modular in nature, enabling the use of a variety of interchangeable representations for storing the sequence index and the annotation matrix. These representations

may be chosen to optimise the space usage for different kinds of inputs and the execution time of desired queries. *MetaGraph* is open source and its code, links to pre-compiled binaries and application examples are available under `https://github.com/ratschlab/metagraph`.

**Scalable multi-sample index construction** The workflow for constructing the *MetaGraph* index consists of three main stages: data pre-processing, graph construction, and annotation.

Typically, the first stage (data pre-processing) involves the distributed construction of separate de Bruijn graphs from the raw data associated with each of the future annotation labels (e.g., the input samples in **Figure 1**, bottom left), and a subsequent cleaning of these graphs to remove possible sequencing errors (**Figure 1**, top left). Annotation labels can be defined to represent any feature of the input sequences, such as sample ID and organism, expression quantile, chromosome number, etc. The graph cleaning step is optional and is not performed on reference data considered to be error-free, such as reference genomes and protein sequences. The curated graphs are then stored as a minimal set of linear paths, called *contigs*, covering all nodes in the graph. This set of contigs acts as a non-redundant representation of the $k$-mers from the original input sequences associated to that annotation label.

In the second stage of construction, all contigs obtained in the first stage are merged into a single joint de Bruijn graph (**Figure 1**, top right). This step can also be carried out in a distributed manner (see Online Methods).

In the third construction stage – graph annotation – all contigs are mapped onto the joint graph to mark the relations of each $k$-mer to the annotation labels. Conceptually, each label forms a column of the annotation matrix and is represented as a compressed sparse binary vector (**Figure 1**, middle right). In an optional additional step, the annotation matrix can be transformed into a representation best suited for the target application (see Section 4.3 for further details).

**Fast and scalable queries on large indexes** As sequence search is a key task for most biomedical analyses, we devised several efficient search algorithms to identify sequence matches in the graph and return associated labels. In the first approach, input sequences are split into $k$-mers, which are directly *mapped* to nodes of the graph. The resulting set of paths directly informs about corresponding annotations (**Figure 2**, top left). For increased sensitivity, we also devised a *graph alignment* algorithm, which identifies the closest matching path in the whole graph or in a sub-graph defined by the annotation columns (**Figure 2**, bottom left; see Section 4.9.2 for details on the method and Section 2.3 for results on accuracy). One important application of this type of query is *experiment discovery*, where each annotation label represents an experiment and the *MetaGraph* index is scanned to find all experiments with reads similar to the queried sequence pattern.

If the query is a single sequence, both mapping and alignment can be applied directly on the full annotated graph. For larger queries of raw read data sets or many longer sequences, we have designed an efficient batch query algorithm (**Figure 2**, right) that is able to exploit the presence of $k$-mers shared between individual queries by forming an intermediate query graph (see Online Methods 4.9.3 and Supplemental Section D.7 for further details). This additional pre-processing leads to a 10 to 100 fold speedup in alignment, depending on the structure of the query data (**Supplemental Figure S-1**).

## 2.2 *MetaGraph* is highly scalable

We evaluate both the construction and the query performance of *MetaGraph* and benchmark them against other state-of-the-art indexing schemes: Mantis [4], BIGSI [12], and COBS [11], using subsamples of increasing size drawn from a large data set of microbial whole genome sequencing samples publicly available in the Sequence Read Archive; this set contains a diverse range of samples consisting of 444,906 virus, Prokaryote, and small Eukaryote genomes, and

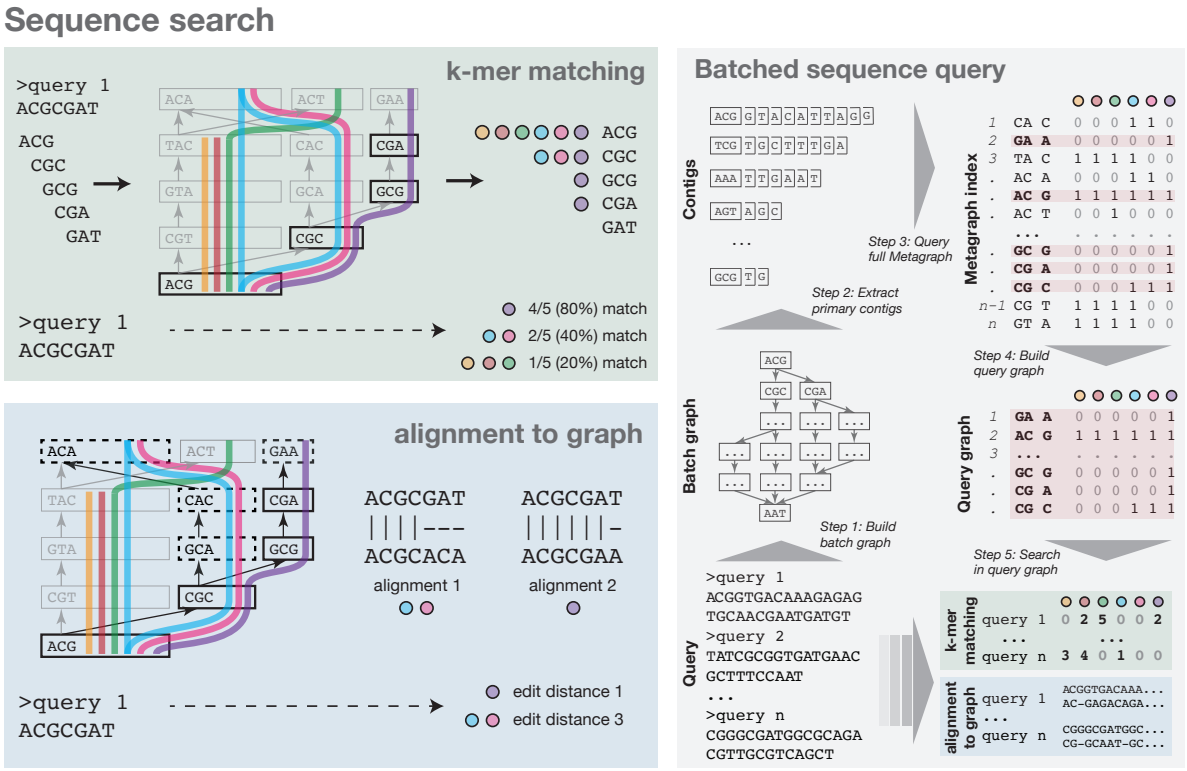


**Figure 2: Graph querying approaches – Left:** Schematic representation of the two main approaches for sequence search. **Top left:** *Ranking* counts exact $k$-mer matches between query and graph. **Bottom left:** *Alignment* finds all closest paths within a given edit distance. **Right:** *Batched sequence search* combines all queries into an intermediate query-graph for subsequent ranking or alignment. All query sequences are assembled into a batch graph that is then traversed to be queried against the full index. Hits are aggregated into the query graph, which is then searched with the original input sequences.

was first presented in the evaluation of BIGSI [12]. Henceforth, we will refer to this data as the *SRA-Microbe* set. A full list of SRA IDs used in the experiments is provided in Supplemental Data Section 5. As all indexing schemes discussed here do compress the input data into a $k$-mer dictionary, the original sequences can only be reconstructed unambiguously if additional information is provided, which makes all of the methods lossy compressors of the input data. In the context of this work, we will focus on the compression of the $k$-mer dictionary itself. While Mantis and *MetaGraph* provide a lossless representation of all $k$-mers, BIGSI and COBS both employ probabilistic data structures that can lead to false-positive matches when the index is queried. Hence, we denote the latter two approaches as *lossy* compressors.

Higher rates of compression often come at the cost of a higher amount of compute for traversal and search. As a consequence, there exists a trade-off between representation size of the index and query performance, as is evident in the Rainbowfish and Multi-BRWT representations for *MetaGraph* (**Figure 3a–c**). In case of probabilistic data structures there is an additional trade-off for accuracy (**Supplemental Figure S-5**). While all representations grow approximately linearly with the input size, the slope and growth behavior are drastically different (**Figure 3a**). For Mantis, the full index grows irregularly with an increasing number of experiments, but is always consistently significantly larger than *MetaGraph* (see also **Supplemental Figure S-5** for further details). Despite using a lossy compression approach, BIGSI and COBS consistently use more memory than *MetaGraph* as well. While BIGSI's memory consumption follows a step function, reflecting a regular doubling of its Bloom filter size, COBS's index growth is closer to linear, but is heavily dependent on the choice of false positive rate. *MetaGraph* outperforms all of the above methods in terms of index size. Even when using the larger, query speed-optimized

annotation formats, *MetaGraph* needs less memory than its competitors (**Supplemental Figure S-5**). Note, that the representation size of an index in memory is not only relevant for construction, but also for any query, as the index needs to be loaded into memory for query operations performed on it. Some key statistics of the SRA-Microbe data set and the final index sizes for *MetaGraph* and BIGSI indexes comprising summary statistics for all data sets presented in this work are listed in **Table 1** and visualized in **Figure 3e**.

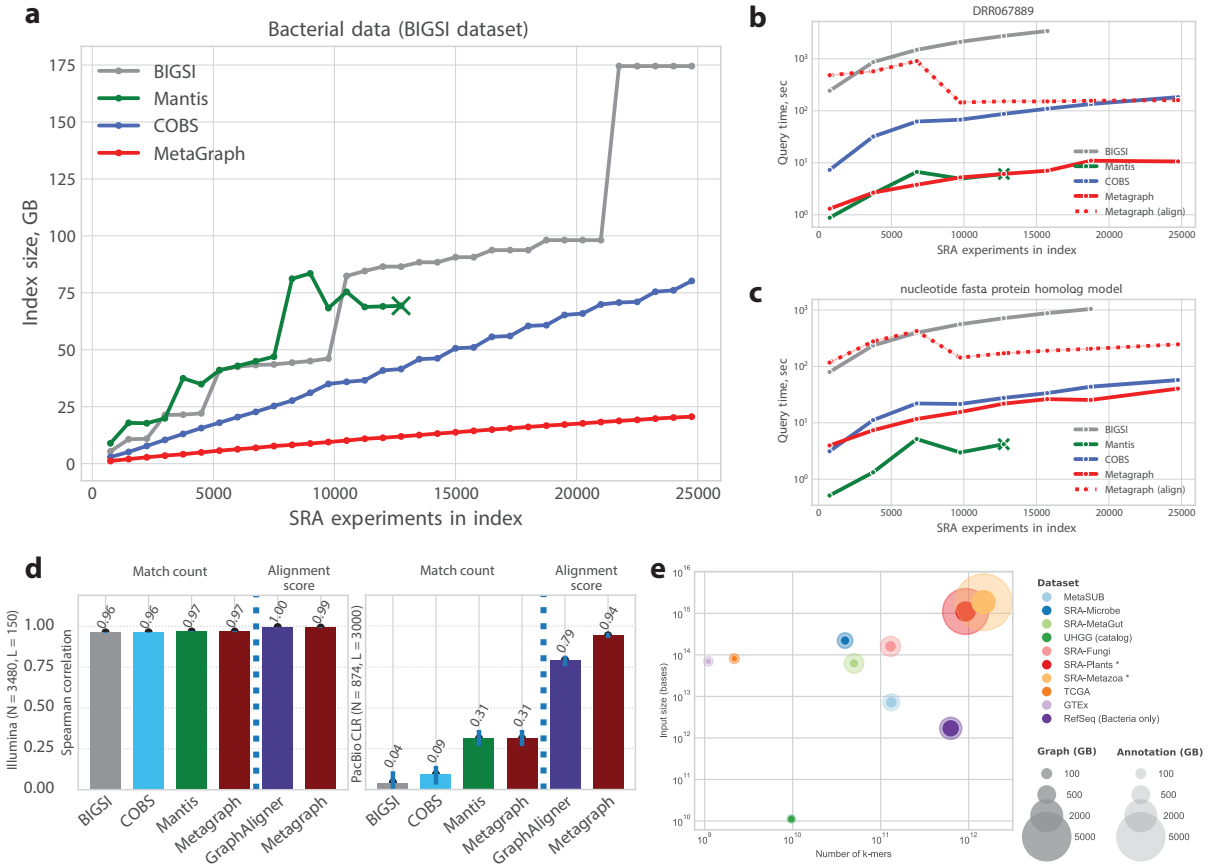


**Figure 3: Scalability and Performance** – **a)** Size of evaluated index data structures for representing a set of bacterial whole genome sequencing experiments of increasing size, shown for both lossless compression methods Mantis (green) and *MetaGraph* (red) and lossy compression methods BIGSI (grey) and COBS (blue). **b)** Query times of the evaluated indexes for querying bacterial genome sequencing data using batched sequence query. Color scheme as in a. **c)** Same as in b, querying AMR gene DNA sequences from the CARD database [1]. **d)** Mapping score correlation to ground truth for different graph approaches and sequencing platforms (left: Illumina, right: PacBio CLR). Bars represent 95% CI's from 100 bootstrap samples of queries. **e)** Overview of all *MetaGraph* index sets presented in this work, showing input size as total number of bases on the y-axis and index size in total number of unique $k$-mers on the x-axis. Marker size indicates the size of the index in GB. The solid portion of each marker represents the fraction of total size taken by the graph and the transparent portion the fraction taken by the annotation.

## 2.3 *MetaGraph* shows superior search performance

Utilizing the SRA-Microbe data set, we tested the query performance of each individual index representation. For our query experiments we only use subsets of up to 30,000 samples, as the memory usage of competing methods exceeded our technical setup.

Using the batch query strategy, *MetaGraph* is able to out-compete all current competitors in query speed up to several orders of magnitude (**Figure 3b** and **c**). Remarkably, this not only

holds for the lossless representations (Mantis), but also for the lossy representations (BIGSI, COBS).

In addition to query performance, we also explored the accuracy of querying against read set-derived genome graphs as a proxy for optimal linear reference sequence alignment (**Figure 3d**, **Supplemental Figure S-2**). For this, we simulated Illumina HiSeq read sets from an *Escherichia coli K-12, MG1655* reference genome (GenBank accession ID `NC_000913.3`) with ART [26] at varying coverage levels and constructed indexes in *MetaGraph*, Mantis, BIGSI, and COBS formats. We then measured the accuracy of both exact match and alignment queries (where applicable) using their respective methods. In addition we measured the accuracy of the sequence-to-graph alignment tool GraphAligner [46] on a GFA representation of the *MetaGraph* index (see Supplementary Section C). For testing, we simulated Illumina HiSeq and PacBio read sets from 12 *E. coli* genomes (at a sum coverage of $1\times$ for each platform) and aligned them to the `NC_000913.3` reference genome with the Parasail aligner [14] to compute ground-truth alignment scores (see Section 4.12.1 for more details). We found that the accuracy of alignment, batch alignment, and $k$-mer matching with *MetaGraph* outperforms all other tools in our comparisons. In addition, we observed that a coverage of $C \geq 20$ was sufficient for the accuracy results to match those obtained by using the original reference to build the graphs (**Supplemental Figure S-2**). Although exact $k$-mer matching did not perform as well for all methods, the alignment methods in *MetaGraph* substantially improve the accuracy of match scores for the simulated PacBio reads.

Although being a compressed representation of the input, the graph structures still show a remarkable sensitivity when queried with the original data, showing that indeed only a small fraction of likely noisy reads was removed during the initial cleaning phase of the graphs. To check how well *MetaGraph* preserves the sequence information present in the raw data and how this affects alignment sensitivity relative to classical linear alignment tools, we re-aligned raw reads from 10 randomly picked GTEx samples [34] back to a graph index containing all GTEx sequences as well as genomic variation present in the gnomAD database [29]. Especially the latter is easy to do using a graph approach, but much harder for classical aligners. When we compared the number of aligned reads between *MetaGraph* and the STAR aligner [16], we found on-par mapping in all cases and superior sensitivity in 8 out of 10 cases (**Supplemental Figure S-3a**). When looking specifically into the fraction of reads unmappable by STAR, we found that generally more than half of these reads can still be aligned by *MetaGraph* and find support in almost all GTEx samples (**Supplemental Figure S-3b**).

## 2.4 Building petabase-scale indexes as a community resource

Applying only moderate cleaning on the input sequences (see Supplementary Section D.4), a *MetaGraph* index typically requires orders-of-magnitude less storage than the original gzip-compressed inputs (**Figure 1e**). The constructed indexes form a valuable community resource, as they succinctly summarise large raw-sequence data sets, while supporting a variety of sequence queries against them. Consequently, we have used the *MetaGraph* framework to construct indexes on real-world data sets of varying size and complexity, including both DNA and RNA sequencing samples (**Table 1**). The range of input data was chosen to represent properties commonly occurring in biomedical research. On the one end stand large cohorts of samples containing sequences sampled from a sequence pool of limited diversity. A representative of this class are the RNA-Seq experiments from the GTEx cohort [34] and the TCGA cohort [52] representing the human transcriptome. With a compressed input size of 40 and 65 Tbp, respectively, the GTEx and TCGA cohorts can be indexed and compressed into annotated graphs of only 157 GB and 65 GB, respectively. In the middle of the complexity spectrum reside whole genome sequencing experiments and collections of reference genomes, where similarity between samples is a function of evolutionary distance and the diversity within the data set is generally much higher than in the human transcriptome cohorts. Representatives of this class are RefSeq

and the UHGG gene genome catalog as well as the SRA-Microbe and SRA-Fungi data sets. While RefSeq contains all sequences of assembled genomes available in version 97 of the RefSeq database, comprising $1.7 \times 10^{11}$ bp of input, the UHGG genome catalog is a recently published catalog of human gut reference genomes [3]. The SRA-Microbe and SRA-Fungi databases are built on 446,506 microbial and 121,907 fungi whole genome sequencing samples, respectively, available on SRA. Each of these datasets represents between 11 gigabases and 221 terabases of sequence input, accumulating to a total of 386 terabases of input. Especially the large-input cohorts, such as SRA-Fungi, required scaling to a commercial compute cloud for the first stage of *MetaGraph* assembly.

As RefSeq covers a much higher diversity of input sequences than, e.g., SRA-Microbe, its index size is correspondingly much larger. The fully searchable and annotated *MetaGraph* representation of the SRA-Microbe data set is only 291 GB (compared to 1.6 TB for the BIGSI index), while RefSeq has a total size of 1,040 GB (when annotated by species taxonomic IDs). Lastly, on the other end of the spectrum stand cohorts containing whole metagenome shotgun sequencing experiments. For this class we have selected the MetaSUB cohort, containing more than 4,200 environmental metagenome sequencing samples comprising 7.3 terabases, and the SRA-MetaGut cohort, containing all human gut metagenome sequencing samples available on SRA ($20,639$ as of 2018-05-25), comprising approximately 60 terabases. The input data in these cohorts are samples from very diverse populations of organisms and contain a large diversity of rare sequences. As a result, the index sizes are relatively large when compared to other data sets, for MetaSUB 315 GB and for SRA-MetaGut 544 GB.

**Table 1:** Summary of constructed indexes for big data sets.

| Dataset | # bp | Input (gz) | k | # $k$-mers | Labels | Graph | Anno. | Ratio |
|---|---|---|---|---|---|---|---|---|
| GTEx [34] | $70 \cdot 10^{12}$ | 40,000 GB | 41 | $26 \cdot 10^{9}$ | 9,786 | 15 GB | 142 GB | 254x |
| TCGA [52] | $81 \cdot 10^{12}$ | 65,000 GB | 31 | $2.1 \cdot 10^{9}$ | 11,095 | 1,6 GB | 63 GB | 1000x |
| RefSeq [40] | $1.7 \cdot 10^{12}$ | 469 GB | 31 | $626 \cdot 10^{9}$ | 48,539 | 339 GB | 410 GB | 0.62x |
| UHGG catalog [3] | $11 \cdot 10^{9}$ | 3,3 GB | 31 | $9.6 \cdot 10^{9}$ | 4,644 | 5,3 GB | 19 GB | 0.13x |
| SRA Microbe [12] | $221 \cdot 10^{12}$ | 170,000 GB | 31 | $39.5 \cdot 10^{9}$ | 446,506 | 31 GB | 264 GB | 576x |
| SRA Fungi | $163 \cdot 10^{12}$ | 81,000 GB | 31 | $277 \cdot 10^{9}$ | 121,907 | 82 GB | 501 GB | 139x |
| MetaSUB [15] | $7 \cdot 10^{12}$ | 5,500 GB | 19 | $71.7 \cdot 10^{9}$ | 4,220 | 49 GB | 266 GB | 17x |
| SRA MetaGut | $63 \cdot 10^{12}$ | 36,000 GB | 31 | $49.9 \cdot 10^{9}$ | 20,639 | 30 GB | 514 GB | 66x |

***MetaGraph* allows for distributed and interactive use** In addition to the single-machine use case, where the graph index is built and queried locally, *MetaGraph* also supports distributed indexes via a client-server architecture. Using this concept, a set of graphs and annotations can be easily distributed across multiple machines. Each machine runs *MetaGraph* in *server* mode, offering one or multiple indexes, awaiting queries on a pre-defined port. This modularity makes it straightforward to integrate one or many queries across a whole set of graphs served on multiple servers. For easy integration of results and coordination of different *MetaGraph* instances, we provide interfaces to popular scripting languages, such as Python, allowing for the interactive usage of one or several (remote) *MetaGraph* index instances (**Figure 4a**).

We demonstrate this usability in publicly available example scripts[1]. In this analysis, the full CARD AMR database [1] is queried against a *MetaGraph* index containing more than 4,400 whole metagenome sequencing samples from the MetaSUB cohort. We use the data to generate a ranking of cities based on the average number of AMR-markers in a sample (**Figure 4b**), largely consistent with the analysis performed on the raw data using orthogonal strategies [15]. Further, the script is used for exploratory analyses, linking sample metadata, such as surface material at the sampling location, to the query results (**Figure 4c**). We invite readers to run the script themselves, reproduce the plots interactively and further explore the available data.

[1] Available at `https://github.com/raetschlab/metagraph`

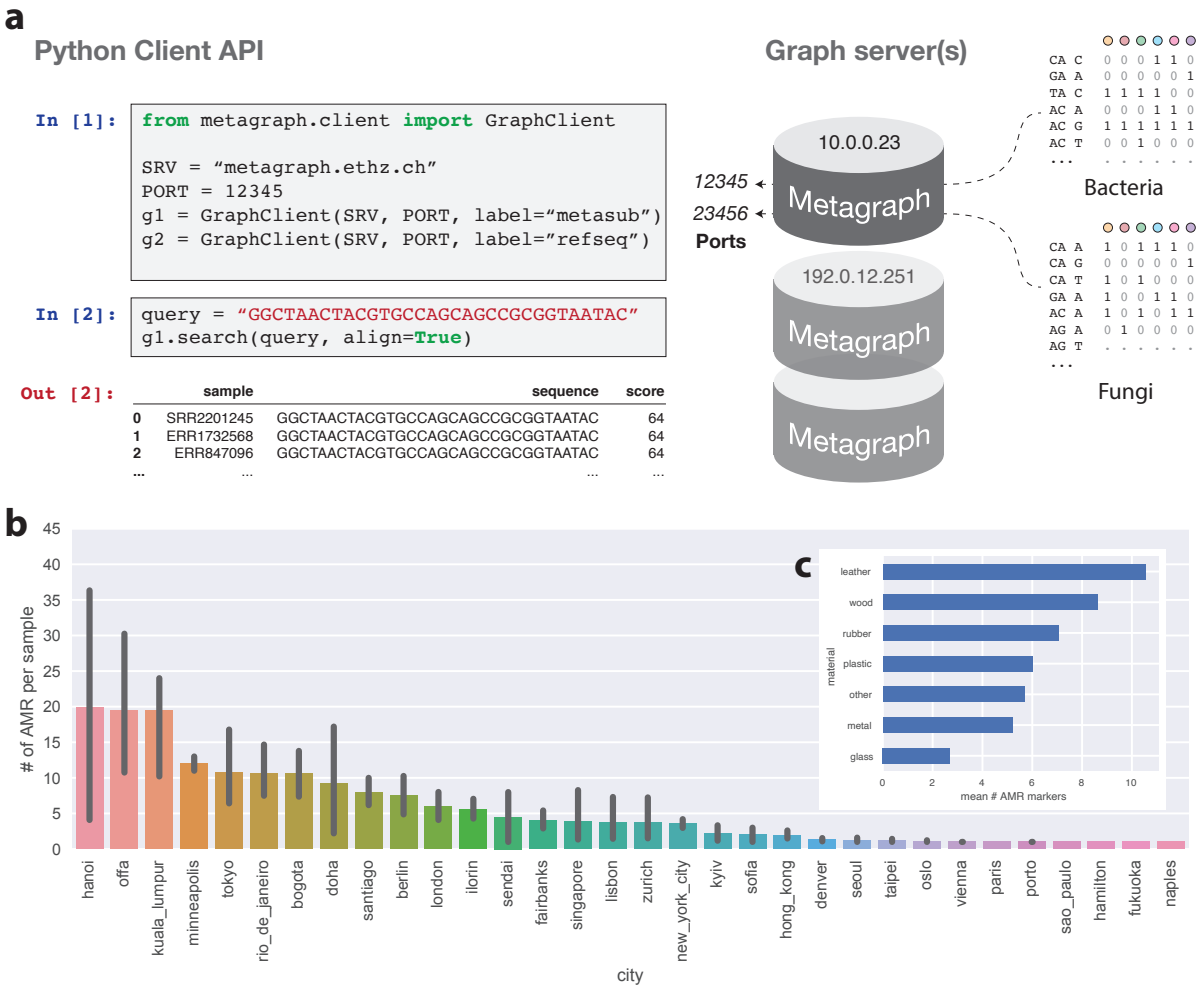


**Figure 4: Utility and usability – a)** *MetaGraph* is designed to support a client-server infrastructure as exemplified here with a Jupyter notebook in Python. In a few steps, several remote (or local) graph instances can be created and queried interactively. Results are returned as a data frame that can be used for further analyses. **b)** Number of antimicrobial resistance (AMR) markers per sample for different cities in the MetaSUB study. Bars represent $\pm\sigma$. **c)** Distribution of the mean number of AMR markers grouped by surface material based on all samples of the MetaSUB dataset.

## 2.5 Differential sequence assembly identifies pathogens in kidney transplant patients

In addition to its sequence search capabilities, the *MetaGraph* index also provides the ability to perform integrative analysis on its samples (corresponding to columnar operations on the annotation matrix; Section 4.10). Conceptually, the individual columns of the annotation matrix form a *node mask* on the graph, implying a sub-graph. Through logical combination of different column sets, sequences can be assembled from these sub-graphs to model differential analyses between groups (**Figure 5a**). We refer to this process as *differential assembly*. For further details, we refer to the Online Methods Section 4.10.

The differential assembly procedure allows for the investigation of interesting biological questions. For instance, given a set of whole metagenome sequencing samples of two patient populations that are distinguished by a certain phenotype (e.g., resistance to treatment), one can categorise the patient samples according to this phenotype and let the annotation columns reflect this categorization (**Figure 5a**, top). The paths identified as differential between the two categories can then be used as markers for further study (**Figure 5a**, bottom). Even in the

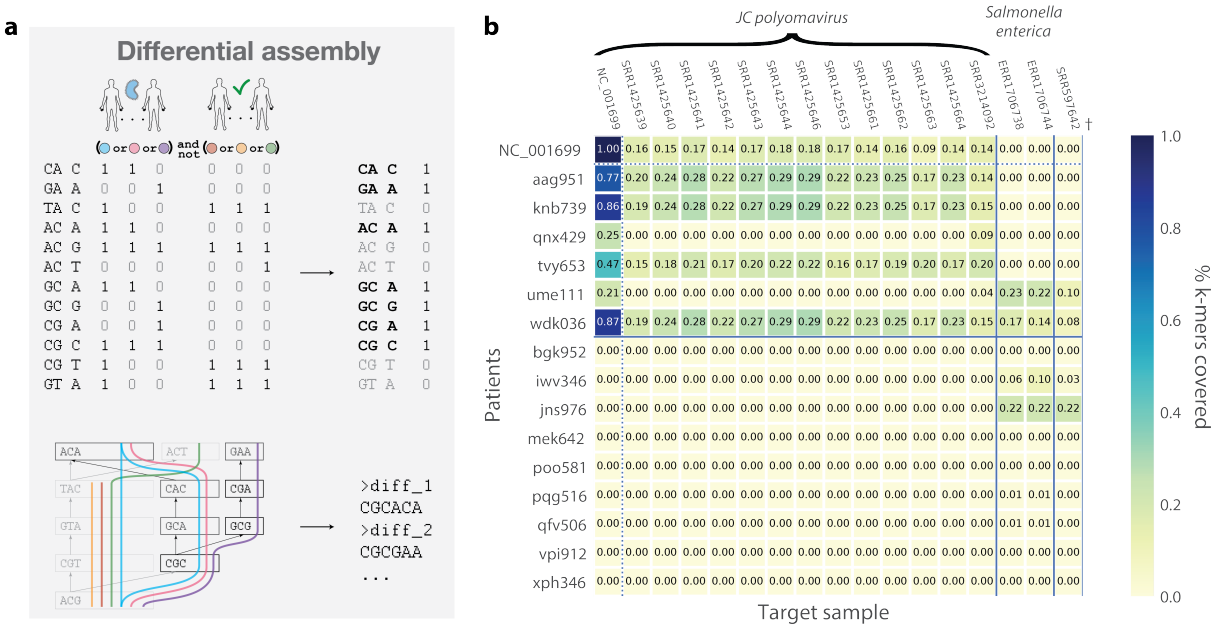


**Figure 5: Differential graph assembly** – **a)** Differential assembly schema. Columns are grouped according to logical operations, forming logical masks that define a subgraph over the full graph. The example shows the subgraph implied by using nodes present in the first three columns, but not present in any of the last three columns. **b)** Assignment of sequences resulting from differential assembly of kidney transplant patients by mapping them to target genomes and samples. Rows represent patient metagenome samples, columns matching labels of the differential assembly. Shading indicates fraction of target genome covered. The `SRR` IDs represent samples annotated as *JC polyomavirus*, *Salmonella enterica*, and uncultured bacteria (†) in the SRA-Microbe index, while `NC_001699` represents the *JC polyomavirus* reference genome.

case that no taxonomic label is known for a sequence associated with the phenotype of interest, *MetaGraph* is still able to identify marker sequences that discriminate between patients and controls.

To illustrate the utility of differential assembly, we performed an analysis on whole metagenome sequencing samples collected from the urine of kidney transplant patients [47]. Using the pre-transplantation samples of donors and recipients as background and the post-transplantation samples of the recipients as foreground sets, we used the *MetaGraph* representation of all samples to assemble sequences present in the foreground but absent in the background for each pair. Although many of the assembled differential sequences is quite short, an appreciable fraction is long enough to be target specific (**Supplemental Figure S-4**). Consistent with the original publication [47], we identified in 6 out of 15 samples *JC polyomavirus* as an agent transmitted with transplantation by mapping the differential contigs to the *MetaGraph* RefSeq index (**Figure 5b**).

## 2.6 Uncovering unexpected transcriptome features in GTEx and TCGA

The Genotype Tissue Expression (GTEx) project has become a *de facto* reference set for human RNA-Seq expression and is widely used in the community [34]. While gene expression values and other summary statistics of the more than 10,000 samples are easily accessible, the whole set of raw sequence RNA-Seq files comprises more than 40 TB even in compressed state. Similarly, The Cancer Genome Atlas (TCGA) has collected more than 10,000 RNA-Seq samples from primary tumors, spanning across more than 30 cancer types, constituting a central resource for cancer research. This cohort also amasses 65 TB in compressed sequences for RNA-Seq samples alone.

For either cohort, tasks that depend on access to the whole data set require an extraordinary effort. For instance, in order to search for the presence of previously unobserved sequence variants in the cohort (e.g., alternative splice forms or somatic variants) the entire data set would need to be re-analyzed.

We compressed 9,759 RNA-Seq samples from GTEx, a total of $70 \cdot 10^{12}$ bases, into a *MetaGraph* index of only 15 GB in size. This index does not only contain all variation detected in the GTEx raw data, but also the human reference genome (version 38) and all variants from the GnomAD cohort (release 3.0) [29]. Adding sample annotation increased the size to 473 GB and 750 GB if quantitative information per sample was used, an approximate 100-fold reduction in size to the original data set. The compression of the TCGA cohort data shows a similar ratio, reducing $81 \cdot 10^{12}$ bases of input data into an annotated graph of 63 GB.

**Detection of trans-splice junctions** Exploiting the efficient query of the graph index, we used it to investigate different features of the transcriptome. First, we assayed the use of exon combinations arising from trans-splicing. In this atypical form of splicing the donor of an exon is not connected to the acceptor of the subsequent exon but to the acceptor of the preceding exon (**Figure 6a**, schema at the top). Due to their non-contiguity, classical linear RNA-Seq aligners are unable to find such alignments, as we illustrate for alignments of a GTEx sample with the STAR aligner [16] (**Figure 6a**, bottom). We created short sequences of length 81 bp spanning all theoretically possible trans-junctions in the GENCODE (version 30) annotation, using the hg38 reference genome. Aligning all sequences not matching to the reference genome to the *MetaGraph* GTEx index resulted in 472 trans junction candidates, perfectly matching against at least one sample path in the graph. Interestingly, the usage of these junction is not uniformly distributed across tissues (**Figure 6b**) but also do not show correlation with the number of samples available per tissue (**Supplemental Figure S-6**). Although having a lower coverage than the flanking regular junctions, the high expression evidence makes an artefactual alignment unlikely (**Figure 6c**).

**Expression evidence for somatic variants in TCGA** We were interested to collect the RNA-Seq expression evidence for a large set of known somatic mutations in the TCGA cohort. Based on all single nucleotide variants of the COSMIC database (version 82), we generated query sequences from the GRCh37 reference genome spanning the variant with additional 20 bp of sequence context both upstream and downstream. All sequences that did not map to the GRCh37 reference, were then aligned to the *MetaGraph* TCGA index. We matched the corresponding variants against the somatic variant calls of the MC3 project, performed on a large set of TCGA samples [18]. For all samples with both expression and MC3 evidence available, we asked which COSMIC variants were both detected in the whole exome sequencing (WXS) based variant calling and supported by RNA-Seq expression evidence. While over half of the positions were confirmed both by RNA-Seq and WXS, showing a Jaccard index of larger than 0.5 (**Figure 6d**), about 30% of the positions were mainly supported by RNA-seq, resulting in a Jaccard index of 0. Thereby especially those positions that were solely found in RNA piqued our interest. One such position is COSM6336980, that expresses the variant allele exclusively in TCGA samples of the thyroid cancer sub-cohort (**Figure 6e**). Our first suspicion of a cancer specific variant was not confirmed, when we found that also all of the normal samples in the THCA cohort expressed the variant allele (**Supplemental Figure S-7**). Interestingly, when confirming in the GTEx *MetaGraph* index, we found that also all GTEX thyroid tissue samples exclusively express the variant allele (**Figure 6f**), which led us to hypothesize tissue-specific RNA-editing as the possible source of this alteration. Interestingly, the observed variant is not a classical A-to-I editing, but instead represents a silent G-to-A alteration, which shows an astonishing pervasiveness across all thyroid tissue samples.

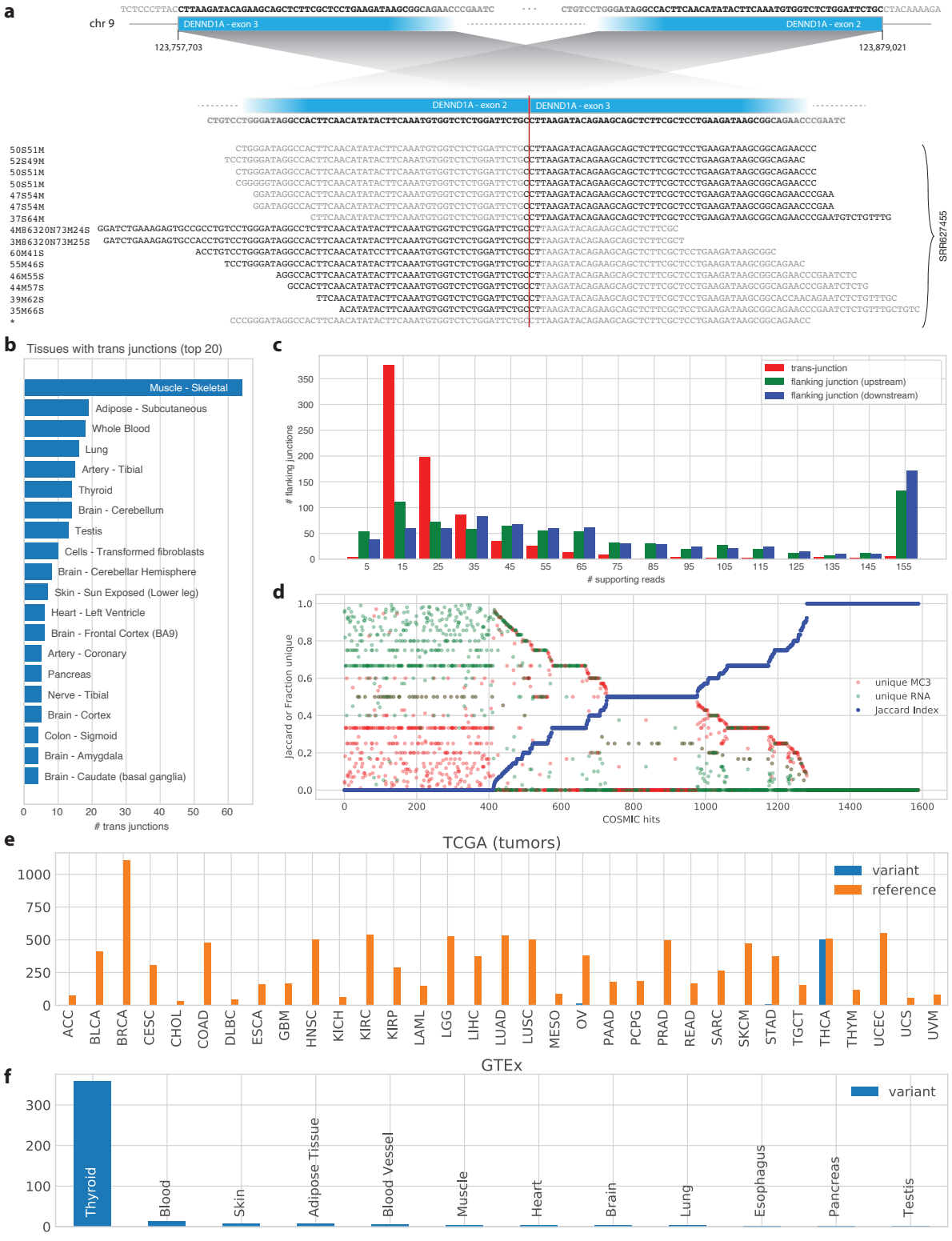


**Figure 6: Transcriptome graphs** – **a)** Schematic of the trans-splicing principle illustrated using the human gene DENND1A (chr 9) as an example. Top: Schematic representation of exon re-arrangement into a trans-junctions. Bottom: Read alignments based on STAR for GTEx sample SRR627455. **b)** Top 20 GTEx tissues sorted by number of detected trans-junctions. **c)** Distribution of the number of supporting reads for all trans-junctions (red), compared to the distribution of read-support for the acceptor of the upstream junction (green) and the donor of the downstream junction (blue). **d)** COSMIC variants in cancer census genes that were found in the *MetaGraph* index. Each value on the x-axis corresponds to a single variant. Variants are sorted by the Jaccard index (blue) describing how well RNA and DNA samples agree on presence of the variant. Number of samples uniquely supporting RNA and DNA is shown in green and red, respectively. **e)** Expression distribution for COSMIC variant COSM6336980 across TCGA tumor samples, grouped by tumor type. Reference allele is shown in orange, variant allele in blue. **f)** Expression distribution for COSMIC variant COSM6336980 across GTEx samples, grouped by tissue. Colors as in e).

# 3 Discussion

Recent advances in DNA sequencing technology have led to massive growth in the amount of high-throughput sequencing data available to the scientific community. However, a lack of standardized approaches for optimal representation and indexing of sequencing data at petabase scale severely limits the interactive exploration of this data and complicates large-scale genomic analysis efforts.

We presented *MetaGraph*, a scalable framework designed for indexing and analysis of large collections of biological sequence data. *MetaGraph* sublinearly scales with the size of the input data and in most cases constructs index representations smaller than those produced by current state-of-the-art methods. In addition, unlike BIGSI and other databases designed for approximate membership queries, *MetaGraph* constructs exact $k$-mer index representations that make no false-positive errors when querying.

We demonstrated the scalability of our approach by constructing queryable indexes for almost the entire collection of microbial, fungi, plant, and metazoa whole genome sequencing datasets present in NCBI's Sequence Read Archive, comprising a total of more than 3.2 petabases of input sequence, representing more than 1.4 million individual samples. We know of no other available method that was applied to such large amounts of input data. Based on our scalability experiments, we expect that the framework can also keep up with future increases in data growth. However, when indexing the available data, we had to restrict ourselves to data generated on platforms with modest error rate. It is part of our future work, to adapt our input cleaning and error correction protocols to also be applicable to sequencing platforms with higher error rates, such as PacBio's SMRT or Oxford Nanopore Technologies long reads.

We have further explored the utility of *MetaGraph* on a number of diverse computational biology applications. Based on a joint cohort of over 20,000 RNA-Seq samples from TCGA and GTEx, we used *MetaGraph* to explore the human transcriptome for interesting features. Thereby our focus was on properties that are hard to detect using linear reference genome alignments or require the integration of many samples to be seen. Specifically, we have evaluated the occurrence of trans-splicing and found over 200 cases commonly occurring within samples of the GTEx cohort. So far our analysis was restricted to trans-junctions occurring within the same gene but could be easily extended into a whole genome assay. A second focus of our transcriptome exploration was on the expression of somatic variants from the COSMIC catalog in different tissue and cancer types. We found that a large fraction of variants has evidence in RNA, but is not reliably called using methods based on whole exome sequencing. One mechanism we suggest contributes to this discrepancy is the occurrence of RNA editing. Thereby we put a special focus on tissue-specific RNA-editing, which seems to occupy a regulatory role. Further investigations in this direction are needed but are out of scope for this work.

Another innovative feature implemented in *MetaGraph* is a generalized framework for sequence assembly from subgraphs. This framework enables the user to fetch biological sequences specific to certain properties or groups of interest (e.g., individual samples, patients subgroups, or any set function of them). As a consequence, *MetaGraph* can answer such queries as "get all sequences found in samples $x$ and $y$ but not present in sample $z$", or "get all sequences shared by all organisms in this taxonomic group". This generalized view on assembly allows for new kinds of integrative analyses. We demonstrated the concept of differential assembly using whole metagenome sequencing of urine samples taken from kidney transplant patients. Using *MetaGraph*, we could reproduce findings on the data in a few minutes and could generalize the analysis. While our sequence assembly methods mirror those of other metagenomics assemblers by extracting unitigs [10, 39, 32] (non-branching stretches of the de Bruijn graph), the problem of chaining together unitigs to form longer contigs and scaffolds is a clear target for future work [9]. In particular, the use of the annotation matrix to motivate the path traversal strategy at branching points could potentially lead to the assembly of longer contigs without compromising assembly quality with heavy cleaning procedures.

A main goal of *MetaGraph* is to make large data sets accessible for interactive exploration. We facilitate this via providing a server-mode for the *MetaGraph*-backend that enables interactive queries from a client via the *MetaGraph* API. Several of the display panels in this work can be reproduced using the public *MetaGraph* instance and a Jupyter Notebook (Supplemental Data, Section 5). The interfaces can be easily extended to also support querying from other languages such as R or Julia.

We envision *MetaGraph* not only to provide a scalable framework for indexing highly diverse sequence databases but also to serve as a versatile tool that enables researchers to perform large-scale comparative analyses in genomics and medicine on typical academic compute clusters. It makes public datasets interactively accessible that are otherwise too big to handle or hard to retrieve. Along with the functionality currently provided, the scalability of many other pipelines could also be improved by translating string matching procedures into equivalent graph operations and supporting such operations using our framework.

# 4 Online Methods

## 4.1 Modularity of the framework

The *MetaGraph* framework consists of two main components: the sequence (k-mer) graph and the annotation matrix. Both parts together form the *MetaGraph Index*. Each component can be transformed into alternative representations, allowing for optimal compatibility to different applications (e.g., per-sequence vs per-label queries). In the following, we will describe the technical details of the graph framework. An explanation of the concepts and on outline of the full construction workflow are described in Section 2.1.

## 4.2 Graph representations

The graph component is fully encoded in a k-mer dictionary. *MetaGraph* supports several data structures to represent this dictionary: i) the succinct de Bruijn graph (*SuccinctDBG*), ii) a compressed bitmap representation (*BitmapDBG*), and iii) a contiguously-stored hash table (*HashDBG*). For further details on the individual representation, please see Supplemental Table S-2 and Supplemental Section D.2.

## 4.3 Annotation representations

Independent of the choice of $k$-mer dictionary representation, a variety of methods are provided for compressing the graph annotation to accommodate for different query types. For fast sequence search queries, we provide implementations of the matrix compression techniques from VARI [37] (*RowFlat*) and *Rainbowfish* [5]. For differential assembly and high compression performance, we provide the Multiary Binary Relation Wavelet Tree (*Multi-BRWT*) compression technique [28]. Finally, two dynamic annotation matrix compressors, *RowCompressed* and *ColumnCompressed* are provided for memory-efficient construction of the annotation matrix. All representations can be converted into each other for adapting compression and query speed to the respective use case.

## 4.4 Graph construction

For construction of the $k$-mer dictionary, all substrings of length $k$ are extracted from the given input sequences. This list of $k$-mers is made unique, where for each k-mer the number of times it occurred in the input is stored. *MetaGraph* provides three choices to generate this so-called k-mer spectrum of the input. First, *MetaGraph* accepts the output of KMC [31] as input. Second, using the `SortedSet` approach, *MetaGraph* can directly generate the k-mer spectrum in memory. Third, in case the k-mer spectrum is larger than the available memory, *MetaGraph*

offers the `SortedSetDisk` approach, which constructs the spectrum in external memory. This last approach has a fixed memory bound, but requires a higher amount of disk I/O. Any of the three steps is then followed by k-mer insertion into the final graph representation.

### 4.5 Sequence extraction/assembly via graph traversal

All sequence information stored in the graph (or a defined subgraph, see Section 4.10) can be extracted and output in FASTA format using the traversal operation. Starting at all nodes with no incoming edges, the graph is fully traversed, writing the corresponding nuleotide of every edge to the output at most once. We require that the resulting set of sequences is a disjoint node cover of the graph (i.e., each $k$-mer in the graph appears in one and only one extracted sequence).

*MetaGraph* distinguishes two main types of traversal output. 1) Traversal in *contig* mode extends the output sequence until no further outgoing edge is present or the next edge has already been output. 2) Traversal in *unitig* mode extends the output sequence until a node with an indegree or outdegree not equal to one is reached, similar to the definition of unitig by [27]. For both traversal modes, we can additionally apply the constraint that only one of the forward and reverse complement representation of a given $k$-mer is output, resulting in *primary $k$-mers*.

For all traversal modes, we assemble sequences in parallel [32] in three stages. We first start from each node with no incoming edges and maintain bit vectors indicating which nodes have been previously visited (see Supplementary Section D.8). We then start traversals from each remaining branching node (i.e., with outdegree $> 1$). Finally, we traverse all remaining simple cycles.

### 4.6 Graph cleaning and refinement

After initial graph assembly, nodes and edges likely originating from noisy inputs are pruned off during contig assembly (see Section 4.10 for more details). For this, *MetaGraph* uses an algorithm developed by Iqbal and colleagues [27]. Briefly, using the $k$-mer spectrum as input, an abundance threshold for solid (non-noise) $k$-mers is estimated. All unitigs shorter than a given length, or where the median read support for $k$-mers in the unitig is below the above threshold, will be pruned off of the graph. For very large cohorts of raw sequencing files, an additional step of pre-filtering based on the $k$-mer spectra can be added, such that all $k$-mers occuring only once (singletons) are discarded.

### 4.7 Dynamic index augmentation and batch updates

Batches of new annotation columns can be added either as a new index constructed and hosted on the same or a different server or they can be merged into an existing index. For merging, the existing index is decomposed into buckets of contigs (each bucket of contigs corresponds to each subgraph induced by the column), then the buckets are extended with the new sequences and the index is constructed from these sequences again. Note that this doesn't require processing the raw data from scratch. We effectively use reduced data which is up to 100 times smaller than the raw unclean reads.

### 4.8 Index distribution scheme

Large indexes can be partitioned into multiple sections (by columns) and hosted on multiple machines. This enables virtually unlimited scalability. The graph representation can also be separated from the annotation representation, generating further flexibility.

## 4.9 Sequence search

This type of query takes a set of sequences as input and results in a corresponding set of paths in the graph and their annotations. For higher throughput, this may be done by simply *mapping* each k-mer in the input sequences to the graph and querying the graph annotation at these node indices to return a label set (see Figure 1 bottom right – Sequence search). For increased sensitivity, each sequence may be *aligned* to the full graph, or to labelled subgraphs to compute their respective closest paths.

### 4.9.1 Exact $k$-mer matching

Each query sequence is decomposed into $k$-mers and these $k$-mers are mapped into the $k$-mer dictionary. Then, the respective rows of the annotation matrix are decoded to answer the query.

### 4.9.2 Sequence-to-graph alignment

The alignment algorithm takes a classic seed-and-extend approach, using several heuristics in both stages to improve query time.

Given an input sequence, seeds are found by finding exact $\ell$-mer matches (where $\ell$ may be less than $k$) and calculating maximal exact matches within the graph's unitigs (called Uni-MEMs) [33].

Each seed is extended forwards and backwards in the graph to produce a local alignment centred at the seed. The extension algorithm is a generalization of the Smith-Waterman-Gotoh local alignment algorithm [22] to de Bruijn graphs, and extends the bit-parallel sequence-to-graph alignment algorithms introduced in [45, 46]. The *MetaGraph* aligner not only features support for affine gap penalties, but also implements heuristics which improve alignment accuracy in the context of long error-prone reads, or graphs derived from low-coverage samples (see Figure 3c).

Briefly, each node in the graph is represented by three score vectors $S$, $E$, and $F$, representing the best alignments so far up to that node (see Supplementary Section D.6 for more details). Similar to Dijkstra's algorithm, this alignment extension step maintains a priority queue of graph nodes whose score vectors have not yet converged in value and iterates until the queue has been exhausted. We additionally employ the X-drop criterion [7, 53] to reduce the search space of the alignment.

This method results in a set of local alignments of the query sequence. From this, a set of non-overlapping local alignments is computed via an algorithm similar to weighted job scheduling (see Supplementary Section D.6 for details).

### 4.9.3 Batched sequence search and alignment

To increase sequence search performance, *MetaGraph* processes query sequences in batches and queries each batch against small *query graphs* extracted from the full index. We introduce a method for constructing query graphs from sequence batches which support both exact match, and accurate alignment search queries.

Given a batch of query sequences, the sequences are transformed into a transient batch graph, which is then traversed to extract non-redundant contigs. The contigs are then queried against the full *MetaGraph* index via exact match. From the hit set, an annotated query graph is constructed representing the drastically smaller intersection of the batch graph with the full *MetaGraph* index. All inputs are then searched against the query graph.

To take advantage of the improved performance of searching against a query graph, while still extracting enough variation to allow for accurate alignment results, the query graph is augmented with an additional set of neighbouring contigs from the full index. We refer to this extension as the *hull* of the initial query graph (see Supplementary Section D.7).

### 4.10 Differential sequence assembly

This second type of query consists in the extraction of sequences from a given subgraph (see Figure 1 bottom middle – Sequence extraction). Depending on how such a subgraph is defined, this may be used for filtering low coverage contigs, constructing the core genome of a cohort, or the enriching for differential sequences in one cohort with respect to a control cohort.

For example, the iterative cleaning steps performed in traditional sequence assembly can be seen as the construction of a *node mask* on a de Bruijn graph, from which contigs are extracted through traversal of the resulting subgraph. Through the use of columnar operations on the annotation matrix, graph masks constructed via the integration of several labels can be used to assemble contigs representing a differential analysis of these samples. We refer to this process as *differential assembly* (Figure 5a). More precisely, a mask containing all $k$-mers present in a foreground cohort of samples which are not present in a background cohort may be defined. Then, during the traversal steps of assembly, nodes which are not included in the mask may be excluded from consideration.

Subgraphs may be selected either based on $k$-mer abundances, or by combining columns from the annotation matrix (corresponding to set operations on the columns' respective $k$-mer sets). In the former case, an initial analysis of the $k$-mer abundaces of the graph is used to determine an abundance cutoff (see Section 4.6). Then, the subgraph is defined as all unitigs whose mean $k$-mer abundace is above this cutoff. In the latter case, given a set of labels, an initial subgraph is defined as all nodes corresponding to these labels. Next, unitigs from this subgraph are extracted and their corresponding annotations are analysed to determine whether they should be included in the final, *refined* subgraph. For instance, given a set of foreground and background labels, a unitig may be kept if the foreground labels are enriched in the unitig's annotations and if the background labels are absent.

### 4.11 API

For a simple and easy to use interaction with the large graph indices, *MetaGraph* offers a Python API that enables programmatic access to a *MetaGraph* server. Started in server mode, the *MetaGraph* index will be persistently present in memory and accepts queries on a pre-defined port. The API then allows sending search or alignment queries to the index and returns the result as a Pandas data frame for further downstream analysis.

### 4.12 Experiments

The following sections briefly summarize the steps taken for the experiments presented in this work.

#### 4.12.1 Measuring the accuracy of sequence search

Given the reference genome for *Escherichia coli K-12 MG1655* (RefSeq accession `NC_000913.3` []), we simulated Illumina HiSeq-type reads with ART Illumina [26] with coverage $C$ and assembled the read set using Metagraph. Then, we constructed Metagraph, BIGSI, and COBS indexes from these contigs. For evaluation, we simulated both Illumina HiSeq and PacBio CLR-type reads (the latter using PBSIM [43]) from 12 *E. coli* reference genomes (see Supplementary Section C) and computed their optimal semi-global alignment scores to the `NC_000913.3` reference using the Parasail aligner [14].

#### 4.12.2 Construction of SRA-Microbe graph

The SRA-Microbe graph was constructed from the same samples used to construct the BIGSI index [12]. A merged canonical graph with $k = 31$ was constructed from cleaned contigs obtained

from the European Bioinformatics Institute FTP file server. The graph was then serialized into primary contigs, then reconstructed and annotated by sample ID to form the final graph.

### 4.13 Construction of the SRA-Fungi, SRA-Plants, and SRA-Metazoa graphs

SRA ID lists of whole genome sequencing experiments from the Fungi, Plants, and Metazoa kingdoms were obtained from the NCBI metadata queried on 09/25/2020, 08/17/2020, and 09/17/2020, respectively, using the Google Cloud Platform's BigQuery service. Briefly, each sample was either transferred and decompressed from NCBI's cloud mirror or downloaded from ENA (if not available on SRA) onto a cloud compute server and subjected to $k$-mer counting using KMC3 [31] to generate the $k$-mer spectrum. If the median $k$-mer count on the spectrum was less than 2, the sample was further processed without cleaning. Otherwise, the sample was subjected to cleaning, using *MetaGraph*'s `clean` mode, pruning tips shorter than $2k$, and using an automatically detected coverage threshold for unitig removal, with a fallback value of 3. After cleaning, for each sample a canonical graph with $k = 31$ was created and serialized into primary contigs. For each cohort (Fungi, Plants, and Metazoa) the serialized samples were joined into a merged graph representation. Further details are available in the Supplemental Methods.

#### 4.13.1 Construction and query of GTEx graphs

All available RNA-Seq samples that were part of the version 7 release of GTEx [34] were downloaded via dbGaP. A list of all samples used is available in the Supplemental Data Section 5. Each sample was individually transformed into a graph using $k = 41$ and then cleaned using *MetaGraph*'s `clean` module, trimming tips shorter than $2k$ and using an automatically detected coverage threshold with fallback 3 for removing noisy unitigs, and then serialised to disk into fasta format. All resulting fasta file were then assembled into a joint graph and then serialized to disk again into primary contigs. From these primary contigs a final graph was assembled. The primary merged graph was then annotated using the cleaned fasta file of each sample, generating one label per sample. All individual annotation columns were then collected into a joint matrix, that was transformed into relaxed Multi-BRWT representation.

#### 4.13.2 Construction and query of TCGA graphs

All available TCGA RNA-Seq samples availabe at the Genomic Data commons were downloaded. A list containing all processed samples is available in the Supplemental Data Section 5. The same assembly and annotation strategy as for GTEx samples was used, with the only difference that $k$ was chosen as 31.

## 5 Supplemental Data

Additional resources for this project, including sample metadata, interactive notebooks and analysis scripts are available in GitHub at `https://github.com/ratschlab/metagraph_paper_resources`. The source code of the *MetaGraph* software is available under GPLv3 License at `https://github.com/ratschlab/metagraph`.

## Acknowledgements

This work was supported through funds of ETH Zurich, MK and HM are funded by the Swiss National Science Foundation Grant No. 407540_167331 "Scalable Genome Graph Data Structures for Metagenomics and Genome Annotation" as part of Swiss National Research Programme (NRP) 75 "Big Data." The authors would like to thank the members of the MetaSUB international consortium for early access to raw sequencing data and their feedback on early versions of

the geolocation DNA sequence search. We also would like to acknowledge the NCBI staff working on maintaining and developing the Sequence Read Archive for their support and interest in our project. The authors thank the members of the Biomedical Informatics Group at ETH Zurich for constant feedback and input on the project. The authors also would like to thank Google Inc. for providing a package of free compute credits on the Google Cloud Platform. We further would like to thank the donors and their families who contributed data to the GTEx and TCGA projects. This study contains data gathered by the Genotype-Tissue Expression (GTEx) Project available through dbGaP at http://www.ncbi.nlm.nih.gov/gap through dbGaP accession number phs000424.v7.p1. This study further contains data gathered by The Cancer Genome Atlas (TCGA) project available through dbGaP at http://www.ncbi.nlm.nih.gov/gap through dbGaP accession number phs000178.v1.p1. The authors are also grateful for the ability to use the publicly available data gathered by the gnomAD project.

# References

[1] Brian P Alcock, Amogelang R Raphenya, Tammy TY Lau, Kara K Tsang, Mégane Bouchard, Arman Edalatmand, William Huynh, Anna-Lisa V Nguyen, Annie A Cheng, Sihan Liu, et al. Card 2020: antibiotic resistome surveillance with the comprehensive antibiotic resistance database. *Nucleic acids research*, 48(D1):D517–D525, 2020.

[2] Bahar Alipanahi, Martin D Muggli, Musa Jundi, Noelle R Noyes, and Christina Boucher. Metagenome snp calling via read colored de bruijn graphs. *Bioinformatics*, 2020.

[3] Alexandre Almeida, Stephen Nayfach, Miguel Boland, Francesco Strozzi, Martin Beracochea, Zhou Jason Shi, Katherine S Pollard, Ekaterina Sakharova, Donovan H Parks, Philip Hugenholtz, et al. A unified catalog of 204,938 reference genomes from the human gut microbiome. *Nature Biotechnology*, pages 1–10, 2020.

[4] Fatemeh Almodaresi, Prashant Pandey, Michael Ferdman, Rob Johnson, and Rob Patro. An efficient, scalable and exact representation of high-dimensional color information enabled via de bruijn graph search. In *International Conference on Research in Computational Molecular Biology*, pages 1–18. Springer, 2019.

[5] Fatemeh Almodaresi, Prashant Pandey, and Rob Patro. Rainbowfish: A Succinct Colored de Bruijn Graph Representation. In Russell Schwartz and Knut Reinert, editors, *17th International Workshop on Algorithms in Bioinformatics (WABI 2017)*, volume 88 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 18:1–18:15, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[6] Stephen F Altschul, Warren Gish, Webb Miller, Eugene W Myers, and David J Lipman. Basic local alignment search tool. *Journal of molecular biology*, 215(3):403–410, 1990.

[7] Stephen F Altschul, Thomas L Madden, Alejandro A Schäffer, Jinghui Zhang, Zheng Zhang, Webb Miller, and David J Lipman. Gapped blast and psi-blast: a new generation of protein database search programs. *Nucleic acids research*, 25(17):3389–3402, 1997.

[8] European Nucleotide Archive. Ena statistics – assembled/annotated sequence growth. `https://www.ebi.ac.uk/ena/about/statistics`. Accessed: 2020-05-26.

[9] Jasmijn A Baaijens, Leen Stougie, and Alexander Schönhuth. Strain-aware assembly of genomes from mixed samples using flow variation graphs. In *International Conference on Research in Computational Molecular Biology*, pages 221–222. Springer, 2020.

[10] Anton Bankevich, Sergey Nurk, Dmitry Antipov, Alexey A Gurevich, Mikhail Dvorkin, Alexander S Kulikov, Valery M Lesin, Sergey I Nikolenko, Son Pham, Andrey D Prjibelski, et al. Spades: a new genome assembly algorithm and its applications to single-cell sequencing. *Journal of computational biology*, 19(5):455–477, 2012.

[11] Timo Bingmann, Phelim Bradley, Florian Gauger, and Zamin Iqbal. Cobs: a compact bit-sliced signature index. In *International Symposium on String Processing and Information Retrieval*, pages 285–303. Springer, 2019.

[12] Phelim Bradley, Henk C den Bakker, Eduardo PC Rocha, Gil McVean, and Zamin Iqbal. Ultrafast search of all deposited bacterial and viral genomic data. *Nature biotechnology*, 37(2):152, 2019.

[13] FP Breitwieser, DN Baker, and Steven L Salzberg. Krakenuniq: confident and fast metagenomics classification using unique k-mer counts. *Genome biology*, 19(1):198, 2018.

[14] Jeff Daily. Parasail: Simd c library for global, semi-global, and local pairwise sequence alignments. *BMC bioinformatics*, 17(1):81, 2016.

[15] David Danko, Daniela Bezdan, Ebrahim Afshinnekoo, Sofia Ahsanuddin, Chandrima Bhattacharya, Daniel J Butler, Kern Rei Chng, Francesca De Filippis, Jochen Hecht, Andre Kahles, et al. Global genetic cartography of urban metagenomes and anti-microbial resistance. *BioRxiv*, page 724526, 2019.

[16] Alexander Dobin, Carrie A Davis, Felix Schlesinger, Jorg Drenkow, Chris Zaleski, Sonali Jha, Philippe Batut, Mark Chaisson, and Thomas R Gingeras. Star: ultrafast universal rna-seq aligner. *Bioinformatics*, 29(1):15–21, 2013.

[17] Richard Durbin. Efficient haplotype matching and storage using the positional burrows–wheeler transform (pbwt). *Bioinformatics*, 30(9):1266–1272, 2014.

[18] Kyle Ellrott, Matthew H Bailey, Gordon Saksena, Kyle R Covington, Cyriac Kandoth, Chip Stewart, Julian Hess, Singer Ma, Kami E Chiotti, Michael McLellan, et al. Scalable open science approach for mutation calling of tumor exomes using multiple genomic pipelines. *Cell systems*, 6(3):271–281, 2018.

[19] Erik Garrison, Jouni Sirén, Adam M Novak, Glenn Hickey, Jordan M Eizenga, Eric T Dawson, William Jones, Shilpa Garg, Charles Markello, Michael F Lin, et al. Variation graph toolkit improves read mapping by representing genetic variation in the reference. *Nature biotechnology*, 36(9):875–879, 2018.

[20] Evangelos Georganas, Aydın Buluç, Jarrod Chapman, Steven Hofmeyr, Chaitanya Aluru, Rob Egan, Leonid Oliker, Daniel Rokhsar, and Katherine Yelick. Hipmer: an extreme-scale de novo genome assembler. In *SC'15: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–11. IEEE, 2015.

[21] Evangelos Georganas, Rob Egan, Steven Hofmeyr, Eugene Goltsman, Bill Arndt, Andrew Tritt, Aydin Buluç, Leonid Oliker, and Katherine Yelick. Extreme scale de novo metagenome assembly. In *SC18: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 122–134. IEEE, 2018.

[22] Osamu Gotoh. An improved algorithm for matching biological sequences. *Journal of molecular biology*, 162(3):705–708, 1982.

[23] Robert S Harris and Paul Medvedev. Improved representation of sequence bloom trees. *Bioinformatics*, 36(3):721–727, 2020.

[24] Glenn Hickey, David Heller, Jean Monlong, Jonas A Sibbesen, Jouni Sirén, Jordan Eizenga, Eric T Dawson, Erik Garrison, Adam M Novak, and Benedict Paten. Genotyping structural variants in pangenome graphs using the vg toolkit. *Genome biology*, 21(1):1–17, 2020.

[25] Guillaume Holley, Roland Wittler, and Jens Stoye. Bloom Filter Trie: an alignment-free and reference-free data structure for pan-genome storage. *Algorithms for Molecular Biology*, 11(1):3, 12 2016.

[26] Weichun Huang, Leping Li, Jason R Myers, and Gabor T Marth. Art: a next-generation sequencing read simulator. *Bioinformatics*, 28(4):593–594, 2012.

[27] Zamin Iqbal, Mario Caccamo, Isaac Turner, Paul Flicek, and Gil McVean. De novo assembly and genotyping of variants using colored de Bruijn graphs. *Nature Genetics*, 2012.

[28] Mikhail Karasikov, Harun Mustafa, Amir Joudaki, Sara Javadzadeh-No, Gunnar Rätsch, and André Kahles. Sparse binary relation representations for genome graph annotation. In *International Conference on Research in Computational Molecular Biology*, pages 120–135. Springer, 2019.

[29] Konrad J Karczewski, Laurent C Francioli, Grace Tiao, Beryl B Cummings, Jessica Alföldi, Qingbo Wang, Ryan L Collins, Kristen M Laricchia, Andrea Ganna, Daniel P Birnbaum, et al. The mutational constraint spectrum quantified from variation in 141,456 humans. *Nature*, 581(7809):434–443, 2020.

[30] Yuichi Kodama, Martin Shumway, and Rasko Leinonen. The sequence read archive: explosive growth of sequencing data. *Nucleic acids research*, 40(D1):D54–D56, 2012.

[31] Marek Kokot, Maciej Długosz, and Sebastian Deorowicz. KMC 3: counting and manipulating k-mer statistics. *Bioinformatics*, 33(17):2759–2761, 2017.

[32] Dinghua Li, Chi-Man Liu, Ruibang Luo, Kunihiko Sadakane, and Tak-Wah Lam. Megahit: an ultra-fast single-node solution for large and complex metagenomics assembly via succinct de bruijn graph. *Bioinformatics*, 31(10):1674–1676, 2015.

[33] Bo Liu, Hongzhe Guo, Michael Brudno, and Yadong Wang. debga: read alignment with de bruijn graph-based seed and extension. *Bioinformatics*, 32(21):3224–3232, 2016.

[34] John Lonsdale, Jeffrey Thomas, Mike Salvatore, Rebecca Phillips, Edmund Lo, Saboor Shad, Richard Hasz, Gary Walters, Fernando Garcia, Nancy Young, et al. The genotype-tissue expression (gtex) project. *Nature genetics*, 45(6):580, 2013.

[35] Charley GP McCarthy and David A Fitzpatrick. Pan-genome analyses of model fungal species. *Microbial genomics*, 5(2), 2019.

[36] Duccio Medini, Claudio Donati, Hervé Tettelin, Vega Masignani, and Rino Rappuoli. The microbial pan-genome. *Current opinion in genetics & development*, 15(6):589–594, 2005.

[37] Martin D. Muggli, Alexander Bowe, Noelle R. Noyes, Paul S. Morley, Keith E. Belk, Robert Raymond, Travis Gagie, Simon J. Puglisi, and Christina Boucher. Succinct colored de Bruijn graphs. *Bioinformatics*, 2017.

[38] Adam M Novak, Erik Garrison, and Benedict Paten. A graph extension of the positional burrows–wheeler transform and its applications. *Algorithms for Molecular Biology*, 12(1):18, 2017.

[39] Sergey Nurk, Dmitry Meleshko, Anton Korobeynikov, and Pavel A Pevzner. metaspades: a new versatile metagenomic assembler. *Genome research*, 27(5):824–834, 2017.

[40] Nuala A O'Leary, Mathew W Wright, J Rodney Brister, Stacy Ciufo, Diana Haddad, Rich McVeigh, Bhanu Rajput, Barbara Robbertse, Brian Smith-White, Danso Ako-Adjei, et al. Reference sequence (refseq) database at ncbi: current status, taxonomic expansion, and functional annotation. *Nucleic acids research*, 44(D1):D733–D745, 2016.

[41] Brian D Ondov, Gabriel J Starrett, Anna Sappington, Aleksandra Kostic, Sergey Koren, Christopher B Buck, and Adam M Phillippy. Mash screen: High-throughput sequence containment estimation for genome discovery. *Genome biology*, 20(1):232, 2019.

[42] Brian D Ondov, Todd J Treangen, Páll Melsted, Adam B Mallonee, Nicholas H Bergman, Sergey Koren, and Adam M Phillippy. Mash: fast genome and metagenome distance estimation using minhash. *Genome biology*, 17(1):132, 2016.

[43] Yukiteru Ono, Kiyoshi Asai, and Michiaki Hamada. Pbsim: Pacbio reads simulator—toward accurate genome assembly. *Bioinformatics*, 29(1):119–121, 2013.

[44] Prashant Pandey, Fatemeh Almodaresi, Michael A Bender, Michael Ferdman, Rob Johnson, and Rob Patro. Mantis: A Fast, Small, and Exact Large-Scale Sequence-Search Index. *Cell Systems*, 7 2018.

[45] Mikko Rautiainen, Veli Mäkinen, and Tobias Marschall. Bit-parallel sequence-to-graph alignment. *Bioinformatics*, 35(19):3599–3607, 2019.

[46] Mikko Rautiainen and Tobias Marschall. Graphaligner: Rapid and versatile sequence-to-graph alignment. *Genome Biology*, 21(1):1–28, 2020.

[47] Peter W Schreiber, Verena Kufner, Kerstin Hübel, Stefan Schmutz, Osvaldo Zagordi, Amandeep Kaur, Cornelia Bayard, Michael Greiner, Andrea Zbinden, Riccarda Capaul, et al. Metagenomic virome sequencing in living donor and recipient kidney transplant pairs revealed jc polyomavirus transmission. *Clinical Infectious Diseases*, 69(6):987–994, 2019.

[48] Brad Solomon and Carl Kingsford. Improved search of large transcriptomic sequencing databases using split sequence bloom trees. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2017.

[49] Brad Solomon and Carl Kingsford. Improved Search of Large Transcriptomic Sequencing Databases Using Split Sequence Bloom Trees. *Journal of Computational Biology*, 25(7):755–765, 7 2018.

[50] Zachary D. Stephens, Skylar Y. Lee, Faraz Faghri, Roy H. Campbell, Chengxiang Zhai, Miles J. Efron, Ravishankar Iyer, Michael C. Schatz, Saurabh Sinha, and Gene E. Robinson. Big data: Astronomical or genomical? *PLoS Biology*, 2015.

[51] Isaac Turner, Kiran V Garimella, Zamin Iqbal, and Gil McVean. Integrating long-range connectivity information into de bruijn graphs. *Bioinformatics*, 34(15):2556–2565, 2018.

[52] John N Weinstein, Eric A Collisson, Gordon B Mills, Kenna R Mills Shaw, Brad A Ozenberger, Kyle Ellrott, Ilya Shmulevich, Chris Sander, Joshua M Stuart, Cancer Genome Atlas Research Network, et al. The cancer genome atlas pan-cancer analysis project. *Nature genetics*, 45(10):1113, 2013.

[53] Zheng Zhang, Scott Schwartz, Lukas Wagner, and Webb Miller. A greedy algorithm for aligning dna sequences. *Journal of Computational biology*, 7(1-2):203–214, 2000.