Lawrence Berkeley National Laboratory
LBL Publications

Title
Deployable high-fidelity metagenome binning at scale with QuickBin

Permalink
https://escholarship.org/uc/item/1pq3s8bd

Journal
Communications Biology, 9(1)

ISSN
2399-3642

Authors
Bushnell, Brian
Villada, Juan C

Publication Date
2026-08-25

DOI
10.1038/s42003-026-10782-z

Copyright Information
This work is made available under the terms of a Creative Commons Attribution License, available at https://creativecommons.org/licenses/by/4.0/

Peer reviewed

eScholarship.org Powered by the California Digital Library
University of California

communications biology

A Nature Portfolio journal

Article

https://doi.org/10.1038/s42003-026-10782-z

Deployable high-fidelity metagenome binning at scale with QuickBin

Check for updates

Brian Bushnell & Juan C. Villada

Reconstructing genomes from metagenomic assemblies is foundational to microbiome research, yet binning faces a persistent trade-off between fidelity and throughput. Many high-accuracy methods rely on GPU-intensive workflows, marker-gene postprocessing, or heavy computational resources, limiting reproducible use at scale. Here, we present QuickBin, a CPU-native, marker-free binning algorithm designed to recover near-complete, ultra-low-contamination metagenome-assembled genomes (MAGs) efficiently. QuickBin pairs a GC-coverage spatial index (BinMap) with an early-exit Oracle cascade of similarity tests (scalar composition/coverage filters and SIMD-accelerated *k*-mer comparisons), reserving a compact neural network exclusively for ambiguous merges. Across synthetic communities, evaluated by marker-based and contig-origin ground truth, QuickBin maximizes high-fidelity sequence recovery. In benchmarking 297 diverse real metagenomes, QuickBin completed all runs, recovering more high-quality MAGs (≥95% completeness, ≤1% contamination) than resource-intensive alternatives that frequently failed. QuickBin provides a practical path to reproducible, genome-resolved metagenomics at scale for downstream comparative analyses. Open-source at: https://github.com/bbushnell/BBTools.

Genome reconstruction from metagenomic data has defined our modern understanding of microbial evolution, metabolism, and ecology by enabling analysis of uncultivated lineages that are present in many environments[1–5]. Metagenome-assembled genomes (MAGs) are widely used in microbiome research to infer functional potential across diverse biomes, from the human gut[6,7] to deep-sea sediments[4] and soil ecosystems[8]. MAGs now support comparative microbiome analyses, ecological inference, and the construction of large genome catalogues, ruled by community quality standards such as the Minimum Information about a Metagenome-Assembled Genome (MIMAG)[1] that formalize expectations for completeness and contamination that enable robust downstream analyses.

A central step in this workflow is metagenomic binning, a process that consists in the assignment of assembled contigs into genome bins using signals such as sequence composition, differential coverage across samples, and assembly-graph connectivity[9]. Over the past decade, a succession of algorithms has improved the resolution of metagenomic binning. Early composition-based methods[10] were succeeded by differential coverage approaches like `MetaBAT2`[11] and `MaxBin2`[12], which remain widely used standards. More recently, the integration of deep learning and variational autoencoders, exemplified by `VAMB`[13], `SemiBin2`[14], and `ComeBin`[15], has further refined binning accuracy by leveraging complex feature embeddings. These advances have facilitated the generation of massive genomic resources, including the Unified Human Gastrointestinal Genome (UHGG) catalog[6], the Genomes from Earth's Microbiomes (GEM) catalog[4], the global species-level repositories[7,16]; with newer bigger catalogs and repositories for MAGs like MGnify[17], gcMeta 2025[5], and SPIRE[18] collectively housing millions of MAGs.

However, the rapid expansion of sequencing throughput has introduced significant computational and quality control challenges. Current binning strategies frequently face a trade-off between scalability and precision[19]. As datasets approach even larger scales, the computational cost of all-versus-all alignment and high-dimensional clustering becomes a bottleneck for existing tools[13,20]. Furthermore, recent quality evaluations of MAGs have revealed that a substantial fraction of publicly available MAGs contain undetected contamination, particularly when automated pipelines prioritize recall over precision[21,22].

Recent foundation-model studies indicate that large-scale pretraining on nucleotide sequences is feasible and can produce representations that transfer to downstream genomic tasks[23–25], while related protein-language-model studies have shown how model behavior depends on the statistical structure of the training data[26,27]. As such efforts expand, the community benefit of genome resources is no longer limited to comparative genomics and ecology. It also extends to enabling reliable genomic AI, where minimizing chimeric artifacts and contamination in training genomes is a reasonable prerequisite for model robustness[28–30]. The imperative for high-fidelity genomic data has intensified with the advent of artificial intelligence

DOE Joint Genome Institute, Lawrence Berkeley National Laboratory, Berkeley, CA, USA. e-mail: bbushnell@lbl.gov; jvillada@lbl.gov

in biology. Genomic foundation models and large language models (LLMs) trained on biological sequences require data of exceptional purity to learn accurate representations of syntax and function[24,31]. Contaminated, fragmented, or marker-gene biased construction of MAGs risk introducing noise that can propagate hallucinations in generative models, or obscure subtle evolutionary signals[28,29,32].

Existing MAG catalogues provide broad coverage, but MAG contamination levels are variable. Growing reliance on MAGs as reusable references elevates the importance of bin purity beyond what is captured by conventional "high-quality" thresholds as MIMAG high-quality genomes allow up to 5% contamination. This motivates development of new tools that aim for high completeness and, especially, low contamination to support use cases that require reliable training data[33–35].

Here, we present QuickBin, a resource-efficient metagenome binning algorithm designed to address scalability and genome-fidelity challenges in large-scale genome-resolved metagenomics. QuickBin leverages a small, pretrained feed-forward neural network, only as a final adjudicator for plausible pairs prioritized by computationally inexpensive heuristics. This enables rapid binning on commodity CPU-only nodes or laptops, on any operating system or architecture, with no compilation or installation dependencies other than Java (or Docker), while prioritizing recovery of near-complete, contamination-free genomes. QuickBin reduces the effective computational burden by combining two ideas. First, it uses BinMap, a GC-coverage spatial index that restricts comparisons to local neighborhoods in feature space rather than performing global all-versus-all evaluations. Second, it uses a hierarchical Oracle decision cascade that aggressively filters candidate merges using inexpensive composition and coverage checks before applying more sensitive SIMD-accelerated k-mer similarity tests; a compact feed-forward neural network is used only as a final adjudicator for the small subset of ambiguous candidate pairs that survive earlier filters. This architecture enables QuickBin to operate robustly on commodity CPU infrastructure with low memory budgets while remaining marker-free, preserving broad applicability across taxa (including Eukaryotes).

We benchmarked QuickBin against widely used and recent binners spanning coverage and composition-based methods such as `MetaBAT2`[11], a variational autoencoder-based binner, `VAMB`[13], the self supervised and contrastive learning binner `SemiBin2`[36], and the contrastive multi view representation learning binner `COMEBin`[15]. We evaluated performance using `CheckM2`[37] quality tiers and ground truth contig origin mapping with `GradeBins`[38]. Across controlled synthetic benchmarks evaluated by `CheckM2` and by contig origin mapping, QuickBin achieved strong genome recovery together with high bin purity, with the clearest advantage in the large multi coverage cases. In real metagenome benchmarks spanning diverse biomes, QuickBin supported robust recovery of high fidelity MAGs in the stringent regime of completeness at least 95% and contamination at most 1%, while maintaining short runtimes and low memory requirements. These results support QuickBin as a practical tool for scalable generation of high confidence MAG sets for comparative microbiome analyses and downstream applications that depend on low artifact genome reconstructions, including genomic machine learning.

Results

QuickBin was developed to enable high-fidelity genome recovery in large-scale metagenomic collections where algorithmic accuracy is often compromised by operational constraints, and to accurately deconvolve high-quality genomes from non-axenic or contaminated isolates and low-complexity commensal libraries. Standard workflows frequently sacrifice bin purity or completeness when assemblies are massive or compute resources are limited. QuickBin addresses these challenges by integrating GC-content, multi-sample coverage, and graph connectivity signals to prioritize conservative contig merging (Fig. 1a). This conservative approach is a prerequisite for recovering genomes that meet beyond MIMAG standards (≤1% contamination), ensuring they are suitable for high-confidence interpretation and the training of genomic foundation models.

The framework's discriminatory power is concentrated in the Oracle cascade (Fig. 1b), a hierarchical decision pipeline designed to reduce chimeric bin formation. To maintain fidelity without exhaustive all-to-all neural network evaluation, the Oracle triages comparisons through a series of early exit filters. It first rejects candidates with divergent GC content, depth ratios, or depth covariance. Candidate pairs that pass these initial filters are then evaluated with hierarchical k-mer cosine similarity tests using trimers, tetramers, and pentamers. The pretrained feed forward neural network is used only as the final adjudicator for the small subset of plausible ambiguous contig pairs that pass all prior heuristic thresholds.

To achieve scalability, QuickBin utilizes BinMap spatial indexing (Fig. 1c). By quantizing GC and multi-sample coverage into a multi-dimensional grid, BinMap constrains candidate comparisons to local neighborhoods. This avoids the quadratic complexity of global comparisons, allowing the algorithm to enforce stringent merge criteria across datasets with millions of contigs while remaining within limited memory and walltime bounds.

QuickBin algorithm

QuickBin solves the computational exhaustion bottleneck in metagenomic binning by prioritizing evidence-based triage. Unlike deep-learning binners that subject every contig pair to expensive neural-network evaluations, QuickBin utilizes a hierarchical Oracle cascade.

This multi-phased approach evaluates sequence composition, coverage, and graph connectivity through a series of early-exit filters. Computationally inexpensive scalars, such as GC content and dimer-derived metrics (HH, CAGA; definitions of HH and CAGA are provided in the "*Derived scalars*" section), are employed first to immediately discard dissimilar pairs. This strategy reserves more sensitive and expensive comparisons, such as trimer frequency cosine difference and the terminal neural network layer, for only the most ambiguous surviving pairs. By combining hierarchical triage with BinMap spatial indexing, QuickBin reduces candidate comparisons while retaining high Clean Bin Bases in controlled synthetic benchmarks and short runtimes in the real metagenome computational benchmark. QuickBin accepts an assembled contig FASTA together with either one or more SAM/BAM alignment files or a coverage cache generated by QuickBin. The quickbin_*contig_depths.txt* cache contains stable contig fields for identifier, index, length, and depth, followed by optional internal graph edge fields generated by QuickBin. All input files can be compressed (gzip, bgzip, etc.). Please refer to the "*Supplementary User Guide*" for instructions on how to run QuickBin.

Calculated metrics

Coverage. Read coverage depth for each contig is calculated from SAM or BAM files, which are loaded via multithreaded streaming then discarded, ensuring memory usage remains independent of the size or number of input alignment files. Coverage can be emitted as a compact text file (`cov.txt`) to allow rerunning without reprocessing SAM/BAM files. Coverage can be input to QuickBin as SAM or BAM, as sorted or unsorted, and the files can be uncompressed or `gzip/bgzip`-compressed. Coverage is calculated as:

$$Coverage = \frac{\#bases\ aligned\ to\ a\ contig}{contig\ length}$$

Coverage covariance. When multiple SAM/BAM files are provided, coverage for each is treated as an independent sample (e.g., a time or depth series) and tracked individually. Coverage cutoffs are then applied on a per-sample basis such that if contigs A and B have similar depth in sample 1 but dissimilar depth in sample 2, they will not be clustered together. Because greater numbers of samples allow greater specificity from coverage covariance alone, with increasing numbers of samples, other cutoffs such as GC difference are relaxed.

To ensure the accuracy of this technique, an entropy calculation determines the number of samples-equivalent of information content. For

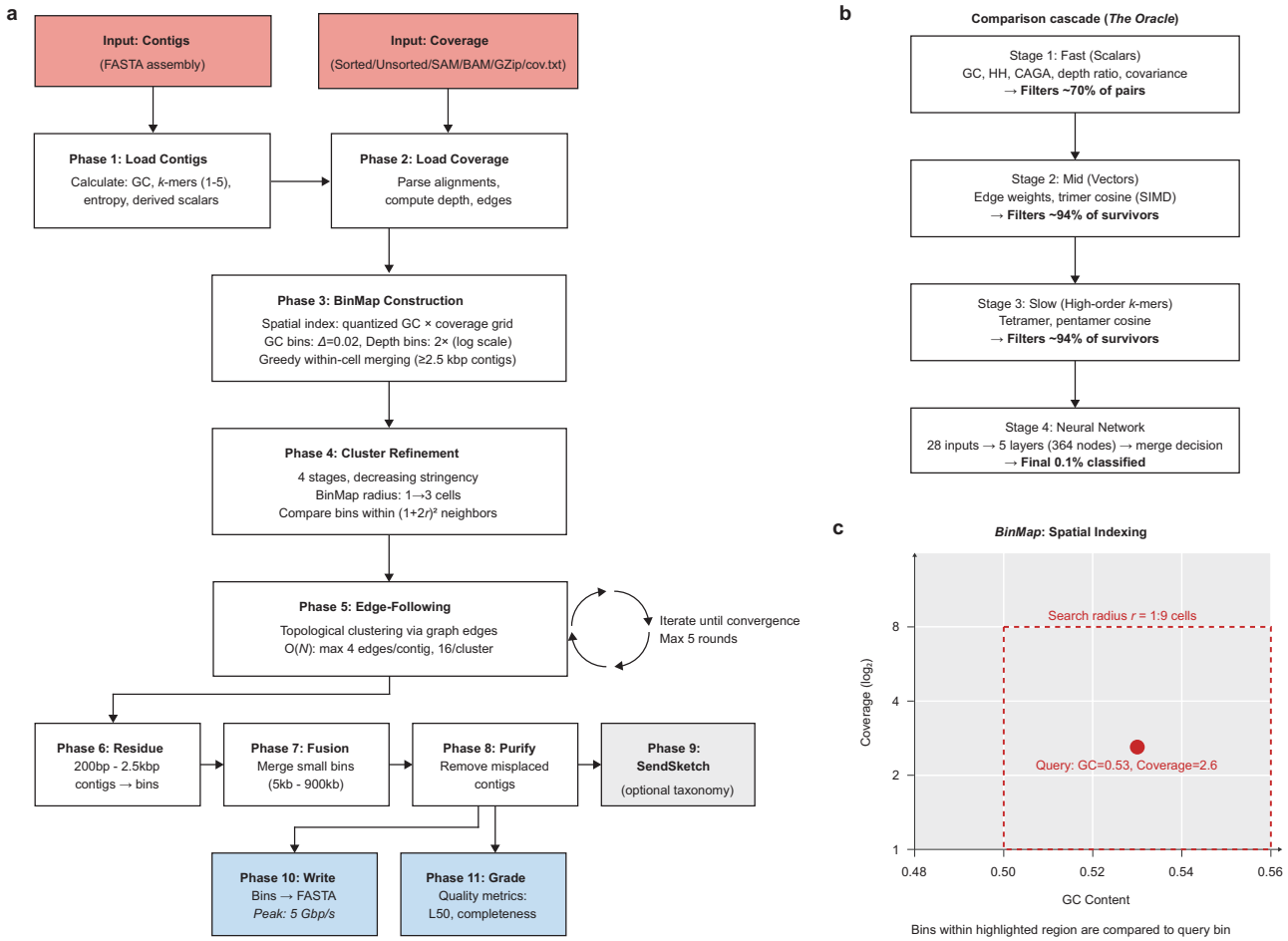


Fig. 1 | QuickBin algorithmic workflow and comparison methodology. a The main algorithm flow through 11 phases. Red boxes indicate input data; white boxes represent processing phases; blue boxes located at the bottom show output generation. Cycle arrows indicate iterative refinement loops. Phases 1-2 load and preprocess input data, computing metrics concurrently via multithreaded pipelines. Phase 3 constructs the BinMap spatial index (see Fig. 1c for details). Phases 4-5 perform primary clustering via spatial (BinMap) and topological (graph edge) similarity, iterating until convergence. Phases 6-8 process small contigs, merge fragmented genomes, and correct greedy algorithm errors. Phases 9-11 handle optional taxonomic annotation, file output, and quality assessment. **b The Oracle comparison cascade**. Bin pairs are filtered through four stages of increasing computational cost: (1) scalar metrics eliminate ~70% of pairs; (2) trimer frequency vectors (SIMD-accelerated) eliminate ~94% of survivors; (3) tetramer/pentamer frequencies eliminate another ~94%; (4) the remaining 0.1% are classified by a neural network trained on synthetic metagenomes. This cascade reduces effective complexity from $O(N^2)$ to near-linear. **c BinMap spatial indexing structure**. The multidimensional index quantizes GC content ($\Delta = 0.02$) and coverage depth (2× logarithmic bins) to partition feature space. The example query bin (red dot; GC = 0.53, coverage=2.6) is compared only to bins in neighboring cells within search radius r. For $r = 1$, this encompasses $(1 + 2r)^2 = 9$ cells (red dashed box), dramatically reducing the number of comparisons from all-pairs $O(N^2)$ to spatially localized subsets. GC guanine-cytosine content, HH homo/heteropolymer ratio, CAGA inter-GC transition affinity, SIMD single instruction multiple data, HQ high-quality, ANI average nucleotide identity, SSU small subunit ribosomal RNA, O(N) computational complexity order notation.

example, the same sample sequenced across two sequencing lanes and presented as 2 SAM files would have their coverage tracked independently but would internally be considered as approximately 1.05 samples of information content due to minor discrepancies in coverage, since they would covary almost perfectly.

GC content:. Calculated as:

$$GC = \frac{(G + C)}{(A + C + G + T)}$$

This is used extensively as an efficient scalar threshold when comparing contigs. GC content is calculated simultaneously with all other short k-mer frequencies during the multithreaded assembly loading phase. As GC content stabilizes toward the genome average with increasing contig lengths, the allowable difference between two contigs is determined by their length, as with other metrics.

Short k-mer frequency:. *k*-mer counts of length 1–5 are calculated during contig loading and stored in `int[]` arrays associated with each contig, as raw counts for $k = 1$–2 and canonical counts for k = 3–5 ($k = 5$ is only calculated for contigs ≥2 kbp). The actual comparisons use a pair of scalars derived from the dimers; SIMD-accelerated cosine difference (*1 − cosine similarity*) for trimers, tetramers, and pentamers; and additionally, GC-compensated cosine difference for tetramers strictly as an input to the neural network. These are stored as counts instead of frequencies to allow fast and exact merging of contigs and clusters with updated count arrays.

The GC compensation calculates frequencies of k-mers relative to only the subset of k-mers with the same GC content; for example, the GC-compensated frequency of AAAA would be calculated as:

$$Freq_comp(AAAA) = \frac{AAAA}{(AAAA + AAAT + AATA + AATT + \ldots + TTTT)}$$

(using only k-mers where GC = 0)

Entropy:. GC-compensated entropy is calculated while loading contigs, by calculating tetramer frequencies over a sliding window of 150 bp and averaging the window entropy over the full contig. The GC compensation was developed using empirical observation of simulated random sequences at varying GC content, such that the range is 0–1 across the GC spectrum rather than extreme GC sequence having a lower maximum entropy and neutral-GC sequence having higher maximum entropy.

Derived scalars:. Strandedness, HH, and CAGA are all derived from dimer frequencies:

- Strandedness: Ratio of sum of minority dimers to the sum of their complements. For example, if |AC| > |TG| and |GC| > |CG|, then AC and GC would be in the denominator, and TG and CG in the numerator; a value near 1 indicates a lack of dimer frequency strand preference (as expected in random sequence), while a lower value indicates bias. This metric is inherently GC-independent.
- HH (Homo/Het Polymer Ratio): Calculated as:

$$\mathrm{HH} = \frac{(AA + CC + GG + TT)}{(AA + TT + AT + TA + CC + GG + CG + GC)}$$

A value over 0.5 indicates that homodimers are more common than heterodimers, amongst dimers with GC = 1 or GC = 0. This metric is inherently GC-independent.

- CAGA (Inter-GC Transition Affinity): Calculated as:

$$CAGA = 0.5 \times \left(1 + \frac{(CA + TG - GA - TC)}{(AC + AG + CA + GA + TC + TG + CT + GT)}\right)$$

Values above 0.5 show a bias toward CA and TG, while values below 0.5 show a bias toward GA and TC. The numerator terms and signs were selected using statistical analysis to determine which dimer frequencies were most strongly correlated or anticorrelated, along with which had the greatest interspecies variance and lowest intraspecies variance. This metric is inherently GC-independent.

These three scalars form GC-independent stable dimensions which are inherent properties of a contig alone, and thus can be calculated independently of coverage or other contigs—unlike GC/Coverage plots or tetramer frequency PCA plots, which rely on external information for dimensions. Of the derived scalars, HH and CAGA differences are used as cutoffs to shortcut more expensive comparisons, while strandedness is only used as a neural network input.

Edges:. Assembly graph connectivity is calculated using paired read mapping, in cases where each read maps to a different contig. These edges are weighted based on the number of spanning pairs compared to the depth of the contig, and depending on how many outward edges a contig has per end. The presence of edges between contigs does not force them into the same cluster, but is used to lower other thresholds and recruit small contigs that would not otherwise be compared at all.

Algorithm phases

Phase 1: Load contigs. QuickBin starts by loading all contigs with a multithreaded file reader; as contigs are read from disk, additional threads competitively calculate contig-specific metrics such as GC, entropy, and k-mer counts. This architecture achieves 590 Mbp/s throughput at an observed 3100% CPU utilization on a 32–core system, loading a 4 Gbp assembly in 6.7 s. Optionally, 16S/18S SSUs are gene-called concurrently. K-mer counting uses an optimized function that counts 1, 2, 3, 4, and 5-mers simultaneously.

Phase 2: Load coverage. QuickBin can be run with or without coverage information. If no alignment information (SAM, BAM, or cov files) is provided, coverage estimates from the assembler will be automatically parsed from contig headers (supporting `Tadpole` [https://bbmap.org/tools/tadpole]), `SPAdes`[39], and `MetaHipMer`[40] headers. In the case of nonuniform amplification such as MDA, coverage is misleading, and the "flat" flag should be used to ignore contig depths, while graph connectivity can still be used based on paired alignment results. However, binning quality will suffer in the absence of coverage information so it should always be used when available and accurate.

When loading SAM or BAM files, multiple files (up to the number of threads) are loaded simultaneously. Each file uses multithreaded `bgzip` decompression and multithreaded file parsing. As a result, bgzipped files (the output of all BBTools when a `.gz` extension is used) are loaded faster than raw uncompressed SAM due to filesystem bandwidth constraints, and even a single BAM file can saturate 13 cores, measured at 1420 Mbp/s. Nevertheless, the coverage and graph connectivity can be stored in a small text file with the "`covout`" flag, which loads almost instantly when reprocessing the same assembly with different parameters. Without a pregenerated coverage text file, coverage loading is usually the longest phase.

Phase 3: BinMap creation. QuickBin represents sequences as Bin objects, with subclasses Contig or Cluster (a mutable set of multiple Contigs). Each Bin has fields such as GC, HH, trimers, and a depth array (per sample) which are compared between bin pairs. During clustering, Contigs or Clusters can be merged to form larger Clusters, causing the values of the new member to be added to the fields of the new or modified Cluster. For example, merging two Contigs would sum the dimer count arrays, perform a size-weighted average of the depth arrays, and force recalculation of GC from monomer counts. Thus, to allow multithreading and prevent problems of synchronization on changing global state, most clustering phases are performed in two subphases: multithreaded comparison of specific Bin pairs, followed by recursive merging of all pairs identified as sufficiently similar. Because this process creates new or modified Bins, the process is typically repeated for a fixed number of rounds or until convergence. The initial creation of the data structure used for most of these phases, though, is singlethreaded, using only a single round and combining comparison and merging greedily.

The BinMap itself is a conceptual GC/cov grid (or with multiple samples, a GC/cov1/cov2 cube) containing a set of lists corresponding to quantized GC or coverage levels. GC is quantized linearly in increments of 0.02 and depth is quantized exponentially by a factor of 2. Thus, a Bin with GC = 0.501, cov = 3.5 would be in the same list as another with GC = 0.505, cov = 3.7; adjacent to a Bin with GC = 0.523, cov = 3.3; and 2 coverage steps plus 1 GC step away from a Bin with GC = 0.492, cov = 9.9.

When creating the BinMap from the input assembly, Contigs are processed in size-descending order and only compared within their own cell. Thus, the first Contig in a cell will be compared to nothing and simply added to the list. Subsequent Contigs will be compared to each existing Bin in the list, and merged with the first one passing the comparison cutoff, or else appended. Therefore, this phase is $O(N^2)$ and would require N(N-1)/2 comparisons if all Contigs were added, no merges were performed, and all Contigs were in the same GC/cov cell. However, in practice it is very fast because elements in the same cell often merge, the metagenomic feature space is well divided by these metrics, and only Bins of at least 2.5 kbp are added to the BinMap.

Phase 4: Cluster refinement. The primary clustering occurs here, with 4 stages of gradually decreasing stringency. Every Bin is compared to all other eligible Bins. Eligibility is decided using the BinMap with a maximal cell distance, which is the number of cells away the two are in the quantized GC or coverage dimension. This range varies from 1 to 3. With range 1, a Bin would be compared to Bins in up to 9 lists: its own list, and all lists with ±0.02 GC or ±1 depth (on a $\log_2$ scale). The number of lists used for comparison is up to $(1 + 2 \times range)^2$, so range 3 uses up to 49 lists of Bins. The comparison is multithreaded, followed by single-threaded merging of identified good pairs to prevent concurrent modification.

Table 1 | Neural network input vector for ground-truth assignment

Category	Feature	Description
Size	$\log_2$(size_a)	Scaled log-transformed length of smaller bin
Size	$\log_2$(size_b)	Scaled log-transformed length of larger bin
Composition	GC_b	GC content of larger bin
Composition	|GC_a − GC_b|	Absolute GC content difference
K-mer similarity	Trimer cosine diff.	Cosine difference of 3-mer frequency vectors
K-mer similarity	Tetramer cosine diff.	Cosine difference of 4-mer frequency vectors
K-mer similarity	Pentamer cosine diff.	Cosine difference of 5-mer frequency vectors
K-mer similarity	Tetramer abs. diff.	Absolute difference of 4-mer frequencies
K-mer similarity	Jensen-Shannon div.	Jensen-Shannon divergence of 4-mer frequencies
K-mer distance	Hellinger (tetramer)	Hellinger distance of 4-mer frequency vectors
K-mer distance	Hellinger (trimer)	Hellinger distance of 3-mer frequency vectors
K-mer distance	Hellinger (pentamer)	Hellinger distance of 5-mer frequency vectors
K-mer composition	GC-comp. tetramer	Cosine diff. of GC-compensated 4-mer frequencies
Probability	K-mer probability	Empirical probability of common origin based on 4-mer difference and contig length
Graph topology	Spanning pairs/depth	Ratio of direct spanning read pairs to mean depth
Graph topology	Trans. pairs/depth	Ratio of transitional (indirect) spanning read pairs to mean depth
Coverage	Depth ratio	Log_2-scaled depth ratio between bins
Coverage	Covariance	Scaled square root of multi-sample coverage covariance
Coverage	Depth_b	Scaled $\log_2(1 + \text{depth})$ of larger bin
Sample count	Samples > 1	1 if multi-sample dataset, 0 otherwise
Sample count	Samples > 2	1 if ≥ 3 coverage samples, 0 otherwise
Sample count	Samples > 3	1 if ≥ 4 coverage samples, 0 otherwise
Sample diversity	Sample entropy	Nonlinear transform of sample count: 1.25·(S−1)/(S+2), where S is the samples-equivalent information content
Sequence complexity	Compressibility_b	8·(1 − entropy_b): high values indicate low-complexity (repetitive) sequence
Sequence complexity	|Compressibility diff.|	8·|entropy_a − entropy_b|: scaled absolute entropy difference
Strandedness	Strandedness_b	Mapping strand bias of larger bin
Strandedness	|Strand_a − Strand_b|	Absolute strandedness difference
Composite	Dissimilarity score	1 − similarity: the heuristic dissimilarity score that would determine the merge decision without the network

Phase 5: Edge-following. If mapped paired reads were provided, or embedded in the cov file, edge-following is performed; all contig pairs with mapped spanning read pairs of sufficient depth are compared to each other, and sufficiently similar pairs are merged into the same cluster. The comparison methodology between pairs is the same in all clustering phases, so the difference between this clustering phase and refinement is simply how the pairs being compared are selected.

For QuickBin, an all-to-all comparison would be close to twice the number of comparisons for all pairs (N(N-1)/2) because it compares not only contig pairs but cluster pairs as they grow. However, this is still $O(N^2)$ rather than $O((2^N)^2)$ (all subsets versus all subsets) since, as a mostly greedy algorithm (adding a contig to a cluster is generally irreversible), each new cluster needs to be compared to at most O(N) elements, and clusters can only be formed or changed O(N) times. The edge-following phase, however, is only O(N) expected, because each Bin compares to at most a constant number of others: 4 for Contigs and 16 for Clusters, using their strongest edges. As with refinement, pairs connected by edges are compared multithreaded, then merges are made singlethreaded. This repeats until convergence or 5 rounds have transpired.

Phase 6: Residue processing. While the initial BinMap construction and refinement was restricted to contigs of at least 2.5 kbp, this phase uses the "residue": all contigs above the "mincontig" cutoff (default 200 bp) but smaller than the BinMap threshold of 2.5 kbp. Each residual contig is compared only to Bins already in the map, not to each other, and added to the best-matching bin according to the Oracle (if it passes the cutoffs). However, since stringency is higher for shorter contigs, most are not added.

Phase 7: Fusion. In an attempt to combine genomes split over a few large bins, fusion compares bins in the size range of 5 kbp–900 kbp (below the expected size of a complete bacterial genome) to larger bins. The algorithm is otherwise the same as refinement apart from lower stringency and size thresholds.

Phase 8: Purification. This is the only subtractive phase and is used to correct errors introduced by greedy merging. Each contig is temporarily removed from its current cluster, and the remaining cluster is recomputed. The removed contig is then compared against the residual cluster using the same Oracle comparison framework used during clustering. If the comparison no longer passes the phase specific similarity threshold, the contig remains removed from that cluster. Each cluster is handled by a single thread during this step to avoid concurrent modification of cluster membership.

Phase 9: SendSketch. Optionally, output Bins are sketched using `BBSketch` (https://bbmap.org/tools/sendsketch.html) and the sketch sent to the `Sketch` server at JGI (https://bbmap.org/services/bbsketch), where it is compared to all RefSeq[41] organisms for taxonomic assignment. This information does not affect binning, but the predicted taxonomy of each bin will be printed in the report file for convenience.

Phase 10: Writing. Bins are now written to disk with a multithreaded writer as a single FASTA file each, using one thread per file. To prevent thrashing on low-performance filesystems, the default number of concurrent files is 1, but throughput increases by 25% when set to 2 on a high-performance flash SSD and over 3× faster when writing to RAM disk (/dev/shm/) with 4 threads, peaking at 5 Gbp/s. If the output files are given a `.gz` extension they will be bgzipped with multiple threads per file. Bins below mincluster (default 50 kbp) are written to a single chaff FASTA file if the chaff flag is set (it defaults to false).

Phase 11: Grading. The completed bins are graded using `GradeBins` (https://bbmap.org/tools/gradebins). Synthetic data with TaxID labels will have completeness and contamination determined for all bins for the report, and a summary such as number of HQ bins and total score will be printed to the screen. For unlabeled data, only statistics such as L50 (size of the 50th percentile bin) and percent sequence recovery are reported.

Comparison methodology. The actual comparison of Bins is the same in all clustering phases, though the stringency varies (generally becoming less stringent in later phases to compensate for the algorithm being greedy). Thus, the main difference between phases is determining which pairs to compare. Comparisons are performed by an Oracle object, which, subject to the desired stringency, makes the merge-or-not decision for a pair of Bins.

First, the GC, HH, and CAGA differences are computed using scalar subtraction. The maximal depth ratio (greater depth/lesser depth) is calculated across all samples. For example, with a single sample, this would simply be:

$$R_depth = \frac{\max(a.\mathrm{cov},\ b.\mathrm{cov})}{\min(a.\mathrm{cov},\ b.\mathrm{cov})}$$

while for 2 samples it would be:

$$R_depth = \max\left(\frac{\max(a.\mathrm{cov1},\ b.\mathrm{cov1})}{\min(a.\mathrm{cov1},\ b.\mathrm{cov1})}, \frac{\max(a.\mathrm{cov2},\ b.\mathrm{cov2})}{\min(a.\mathrm{cov2},\ b.\mathrm{cov2})}\right)$$

so it is selected from the sample in which their depth differs the most. To prevent extreme ratios in low-depth contigs, a small constant is added to the numerator and denominator. If there are multiple samples, the coverage covariance is also calculated using the cosine difference of the normalized depth arrays. At this point, any of these metrics failing a cutoff will trigger an early exit from the comparison; this is called a "fast comparison" and typically eliminates approximately 70% of pairs from further consideration with just a few scalar operations. Note that most comparisons are never performed in the first place due to spatial indexing via the BinMap.

Second, graph edges between the pair are detected to adjust subsequent cutoffs; the cutoffs are less stringent for bins sharing edges. Some cutoffs are reevaluated in the presence of the additional connectivity information, and the cosine difference of the canonical trimer frequencies is calculated and subjected to its own cutoff. All cosine difference calculations are accelerated using 256-bit SIMD if available. This stage, "mid comparison", eliminates approximately 94% of the remaining pairs.

Third, tetramer and (if both bins are over 40 kbp) pentamer frequencies are calculated and subjected to cutoffs. This stage, "slow comparison", eliminates another 94% of the remaining pairs.

Finally, when the neural network is used, the surviving 0.1% of comparisons have the final decision made by the network. Otherwise the final decision is made using a composite of all metrics with a different cutoff. The network is a sparsely connected feed-forward network with 75% connection density with 28 inputs and 5 hidden layers, containing 364 nodes and approximately 25,000 weights, using mostly sigmoid and tanh activation functions with custom rslog (rotationally symmetric logarithm) as the final layer to better handle diminishing gradients. The network was trained using BBTools' train.sh (https://bbmap.org/tools/train), using synthetic data from Tadpole-assembled TaxID-labeled contigs, coverage generated with RandomReadsMG[42], and AllToAllVectorMaker (makequickbinvector.sh [https://bbmap.org/tools/makequickbinvector]) which makes random subsets of contigs to compare for the purpose of making tsv files of vectors. The details of the input vector are listed in Table 1.

While QuickBin can be run without using the network, it dramatically improves the results on synthetic data (where the answer is known). See sections on "*Benchmarking* QuickBin *on synthetic data*" for more details.

In SSU mode (with the "ssu" flag), the final decision of merge/not merge can be vetoed when two bins passing all comparison cutoffs each have an identified SSU (typically 16S sequence, but 18S is also used). These sequences are aligned with the fast traceback-free QuantumAligner, and the merge is rejected if the ANI is below 96%.

Selection of state-of-the-art binners for performance benchmarking

To establish a robust baseline for evaluating QuickBin, we first assessed the current landscape of metagenomic binning tools using performance metrics recently synthesized from a comprehensive benchmark by Han et al.[22]. An analysis of overall quality versus computational efficiency reveals a sharp trade-off between reconstruction quality and resource requirements of state-of-the-art algorithms (Supplementary Fig. 1). Deep learning-based methods, specifically `COMEBin`[15] and `SemiBin2`[36], currently set the standard for reconstruction quality, ranking first and third, respectively. However, this precision comes at a substantial computational cost; `COMEBin` ranks lowest (13th) in both RAM and time efficiency, while `SemiBin2` similarly ranks in the bottom tier for computational speed. `MetaBinner`[20] was not considered, as `COMEBin` was presented by the same group of authors as an improved tool.

Widely adopted tools such as `VAMB` and `MetaBAT2` showed strong resource efficiency in recent binner benchmarks[22,43], whereas `COMEBin` and `SemiBin2` represented high recovery modern methods with higher computational cost. This tradeoff informed our direct benchmark set. We selected `COMEBin` and `SemiBin2` to represent recent SCG marker based and learning based methods with strong reported recovery, and `MetaBAT2` and `VAMB` to represent widely used efficient binners. `MaxBin2` was considered in the external ranking summary, but was not selected for direct benchmarking because it ranked 10th in quality, 4th in RAM efficiency, and 9th in runtime (Supplementary Fig. 1); also, `MetaBAT2` provided the more competitive traditional coverage and composition comparator in both performance and scalability. We therefore chose to compare QuickBin against COMEBin and SemiBin2 to assess maximum achievable precision, and against MetaBAT2 and VAMB to evaluate computational performance at scale. Software versions and database details used for the present study are provided in Supplementary Table S1.

JGI bacterial isolates define a high-fidelity baseline for genomic reconstruction

To empirically determine rigorous quality thresholds for MAG recovery, we established a baseline of "gold standard" genomic fidelity using a reference set of bacterial isolates sequenced and curated by the Joint Genome Institute (JGI). We analyzed 15,554 JGI bacterial isolates that had undergone extensive Quality Assurance and Quality Control (QA/QC) to assess their completeness and contamination profiles using `CheckM2`. This analysis shows that high-quality reference isolates exhibit near-perfect completeness and negligible contamination. The mean quality metrics for this dataset were calculated at 99.82% completeness and 0.66% contamination (Supplementary Fig. 2). These values situate the average isolate well within the ultra-high-quality spectrum (Contamination < 1%). This empirical baseline shows that while maximizing completeness is critical, minimizing contamination to sub-1% levels is equally necessary to approximate the fidelity of isolate-derived genomes for uses such as training robust genomic foundation models.

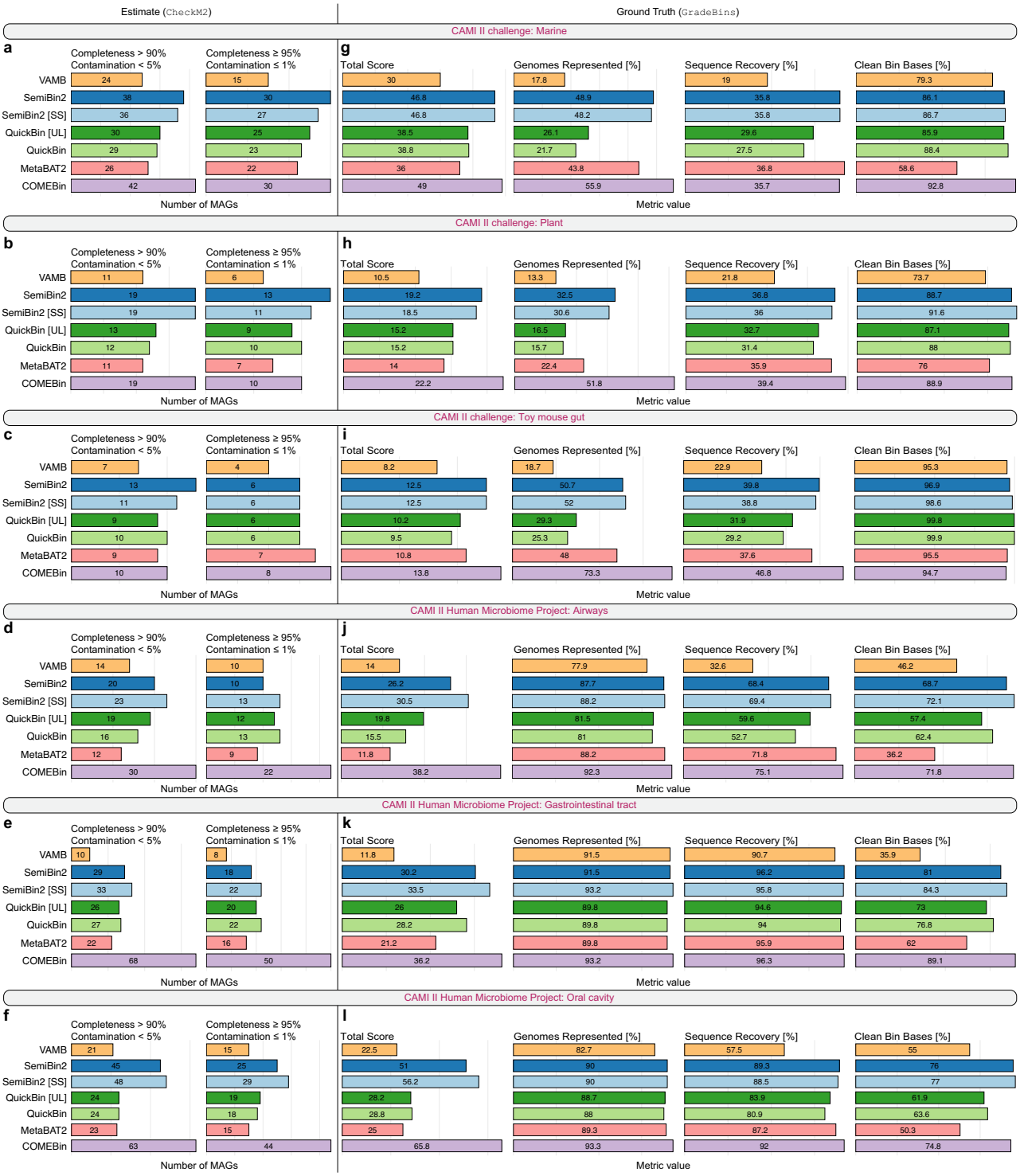


Fig. 2 | Benchmarking of binners on CAMI II metagenomes by CheckM2 quality estimates and contig origin (ground truth) mapping with GradeBins. `CheckM2` quality estimates for CAMI II marine, plant, toy mouse gut and Human Microbiome Project body site panels. Panels show marine in **a** plant in **b** toy mouse gut in **c** airways in **d** gastrointestinal tract in **e**, and oral cavity in (**f**). Bars report the number of MAGs passing the two displayed quality tiers, greater than 90% completeness with less than 5% contamination, and at least 95% completeness with at most 1% contamination. `QuickBin` [UL] indicates ultraloose mode. `SemiBin2` [SS] indicates self supervised mode. `GradeBins` scoring of the same CAMI II panels using contig origin truth. Panels show marine in **g** plant in **h** toy mouse gut in **i** airways in **j** gastrointestinal tract in **k** and oral cavity in **l**. Bars report Total Score, Genomes Represented, Sequence Recovery and Clean Bin Bases. Ground truth metrics calculation used `GradeBins` with CAMI genome identifiers as source labels. Higher values indicate greater total recovery, broader genome representation, greater sequence recovery, or cleaner bin assignment for the corresponding metric. Total Score is the composite `GradeBins` summary and should be interpreted as an overall ground truth performance score, not as the number of MAGs recovered.

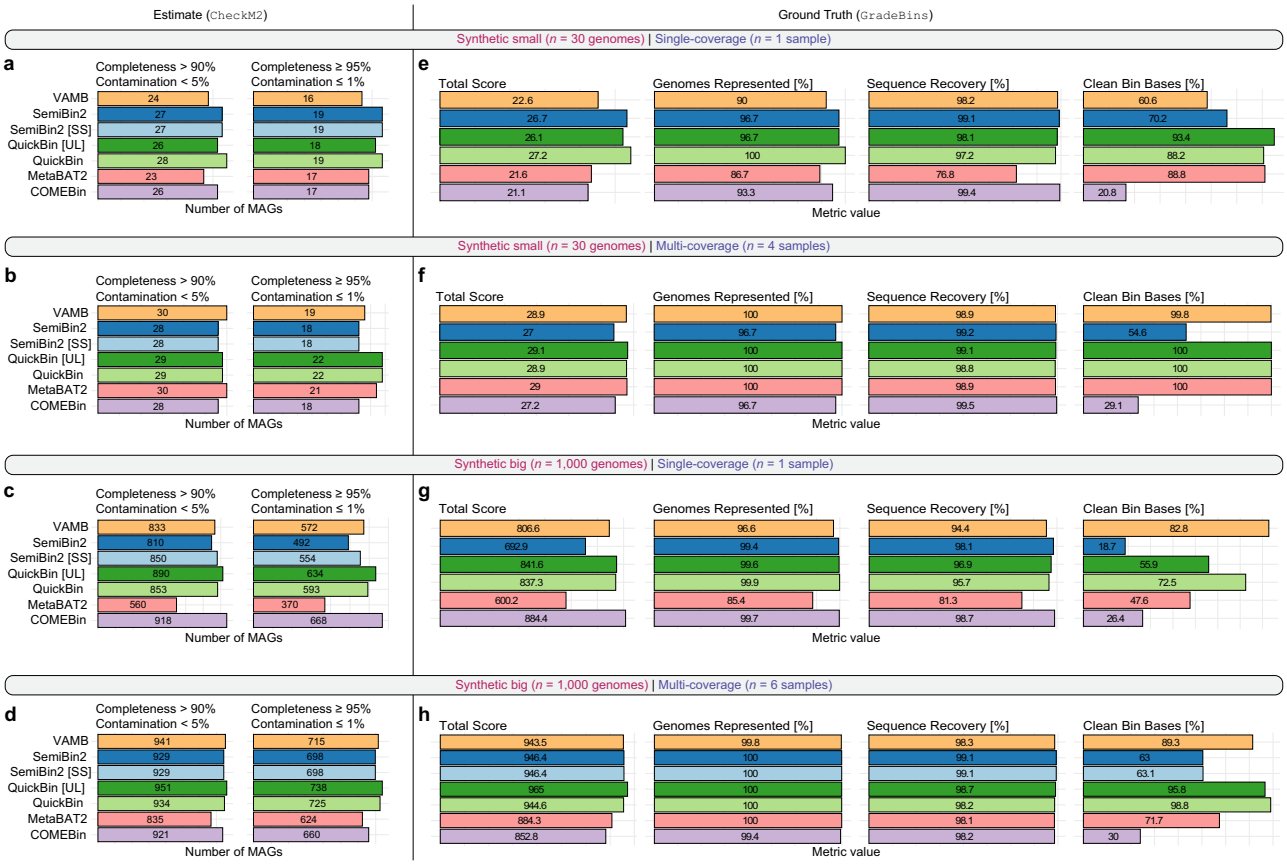


Fig. 3 | Benchmarking of binners on controlled synthetic metagenomes by CheckM2 quality estimates and contig origin (ground truth) mapping with GradeBins. `CheckM2` quality estimates for `RandomReadsMG` synthetic communities with controlled genome number and coverage design. Panels show the synthetic small community with 30 genomes under single coverage with one sample in **a** and multi coverage with four samples in **b** and the synthetic big community with 1000 genomes under single coverage with one sample in **c** and multi coverage with six samples in (**d**). Bars report the number of MAGs passing the two displayed quality tiers, greater than 90% completeness with less than 5% contamination, and at least 95% completeness with at most 1% contamination. `QuickBin` [UL] indicates ultraloose mode. `SemiBin2` [SS] indicates self supervised mode. `GradeBins` scoring of the same controlled synthetic communities using contig origin truth. Panels show the synthetic small community under single coverage in **e** and multi coverage in **f** and the synthetic big community under single coverage in g and multi coverage in (**h**). Bars report Total Score, Genomes Represented, Sequence Recovery, and Clean Bin Bases. Total Score is the composite GradeBins summary and should be interpreted as an overall ground truth performance score, not as the number of MAGs recovered. Ground truth metrics calculation used `GradeBins` with synthetic contig origin labels as the source labels. Higher values indicate greater total recovery, broader genome representation, greater sequence recovery, or cleaner bin assignment for the corresponding metric.

CAMI II benchmarks reveal metric dependent binner rankings and competitive clean bin recovery by QuickBin

We next benchmarked QuickBin on CAMI II metagenomes, where contig origin truth enables direct evaluation of recovery and purity. QuickBin and QuickBin [UL] were compared with `VAMB`, `SemiBin2`, `SemiBin2` [SS], `MetaBAT2` and `COMEBin` across marine, plant, toy mouse gut and Human Microbiome Project body site communities. Each binner was evaluated using `CheckM2` quality estimates and by `GradeBins`[38] scoring against CAMI genome identifiers.

`CheckM2` estimates showed that estimated MAG yield depended strongly on both community type and binner. `COMEBin` recovered the largest estimated MAG sets in the marine challenge, whereas `SemiBin2` was strongest in the plant challenge and several methods performed similarly in toy mouse gut (Fig. 2a–c). In the Human Microbiome Project body site panels, `COMEBin` produced the largest estimated MAG sets, while QuickBin recovered smaller but consistent sets across airways, gastrointestinal tract, and oral cavity (Fig. 2d–f). These results show that `CheckM2` based ranking was variable across CAMI II community types, but overall results favored SCG marker-based binners like `SemiBin2` and `COMEBin`.

Contig origin scoring separated number of MAGs recovered from bin purity. `COMEBin` achieved the strongest Total Score across the CAMI II panels, consistent with broad genome representation and high sequence recovery. However, Clean Bin Bases identified a distinct performance axis. QuickBin retained competitive clean bin recovery in the marine and plant challenges, and produced the cleanest bins in toy mouse gut despite lower number of MAGs recovered than the highest recovery methods (Fig. 2g–i). This pattern indicates that greater sequence recovery did not always correspond to cleaner bin assignments.

In the Human Microbiome Project metagenomes, `GradeBins` scoring more clearly favored `COMEBin` and both `SemiBin2` modes, particularly for Total Score, genome representation, and sequence recovery (Fig. 2j–l). QuickBin remained competitive in selected recovery metrics, especially in gastrointestinal tract, but did not lead clean bin recovery across these host associated communities (Fig. 2j–l). Overall, the CAMI II benchmark shows that binner rankings are metric dependent. QuickBin provides competitive clean bin recovery in several CAMI II settings, but its relative advantage depends on community structure and on whether the endpoint prioritizes estimated MAG yield, genome breadth, sequence recovery, or clean base purity.

Controlled synthetic benchmarks show that QuickBin balances genome recovery with bin purity

CAMI II provided a benchmark with contig origin truth, but it did not isolate how community size and coverage design affect binner performance.

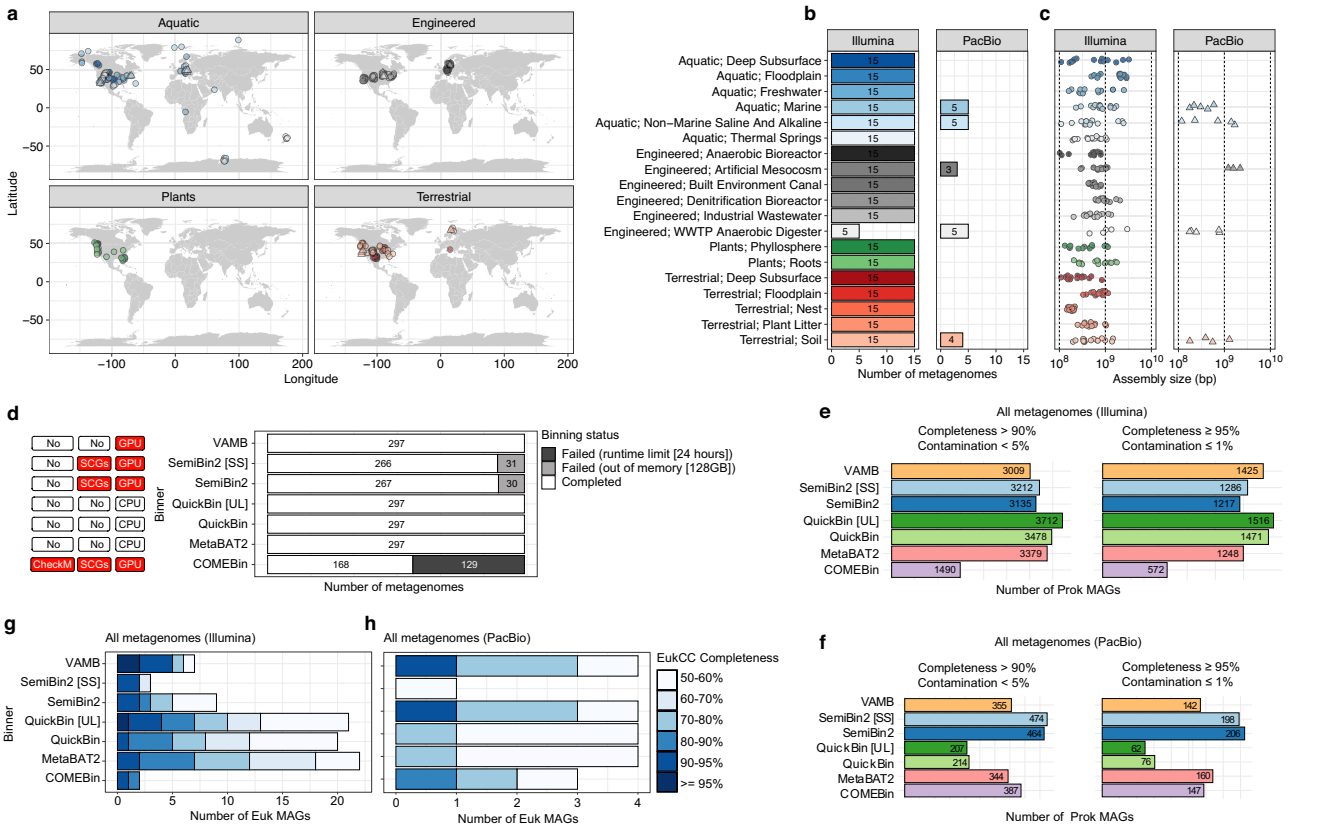


Fig. 4 | Benchmarking of binners on real metagenomes by completion status, CheckM2 quality estimates and EukCC completeness. Composition of the real metagenome benchmark. **a** Shows sampling locations for Aquatic, Engineered, Plants and Terrestrial metagenomes. **b** Shows environmental subtype counts split by sequencing platform. Panel **c** shows assembly size distributions for Illumina and PacBio assemblies. **d** Binner feature and completion summary for 297 metagenomes. The matrix reports whether each binner depends on `CheckM1`, SCG markers, or GPU hardware. Red boxes indicate dependency. The completion bar plot reports completed and incomplete runs under the standardized resource caps. Incomplete runs correspond to wall time limit or memory exhaustion. `QuickBin`, `QuickBin` [UL], `MetaBAT2`, and `VAMB` completed all samples, whereas `COMEBin` and both `SemiBin2` modes had incomplete runs. `COMEBin` was the only evaluated binner without a user configurable minimum contig length for binning. `COMEBin` was therefore run with default settings, whereas all other binners used a 1.5 kbp minimum contig length. `CheckM2` quality estimates for prokaryotic MAGs recovered from real metagenomes. **e** Shows Illumina metagenomes, and **f** shows PacBio metagenomes. Bars report the number of MAGs passing the two displayed quality tiers, greater than 90% completeness with less than 5% contamination, and at least 95% completeness with at most 1% contamination. The stricter tier is used in this manuscript as the high fidelity MAG target. `QuickBin` [UL] indicates ultraloose mode. `SemiBin2` [SS] indicates self supervised mode. `EukCC` completeness estimates for eukaryotic MAGs recovered at `EukCC` estimated contamination less than 10%. **g** Shows Illumina metagenomes and **h** shows PacBio metagenomes. Bars are stratified by `EukCC` completeness bins. Higher counts indicate greater eukMAG recovery in the corresponding completeness tier.

We therefore generated controlled synthetic metagenomes from an independent set of source genomes using `RandomReadsMG`. This design allowed us to vary community size and coverage structure systematically, with small and large communities evaluated under single coverage and multi-coverage inputs. The goal was to determine whether differences among binners reflected estimated MAG yield, sequence recovery, bin purity, or the added information provided by multiple coverage profiles. We evaluated each output using `CheckM2` quality estimates and by `GradeBins` scoring against synthetic contig origin labels.

`CheckM2` estimates showed that QuickBin was consistently among the strongest methods, but the leading binner changed with community size and coverage structure. In the small community, QuickBin was part of the leading group under single coverage, and both QuickBin modes produced the largest strict tier MAG yield under multi coverage (Fig. 3a, b). In the large community, `COMEBin` and the QuickBin modes led under single coverage, whereas QuickBin and QuickBin [UL] led the strict tier under multi coverage (Fig. 3c, d). These results indicate that multiple coverage profiles increased the relative advantage of QuickBin, especially when community complexity was high.

Contig origin scoring separated number of MAGs recovered from bin purity. In the small single coverage benchmark, `QuickBin` achieved the strongest overall ground truth performance and complete genome representation while retaining high Clean Bin Bases (Fig. 3e). In the small multi coverage benchmark, `QuickBin` and `QuickBin` [UL] recovered the full community with near complete sequence recovery and complete clean bin recovery (Fig. 3f). Other methods also recovered broad sequences in this setting, but lower Clean Bin Bases for some binners showed that high sequence recovery did not always correspond to clean bin assignment (Fig. 3f).

The large community made the recovery and purity tradeoff clearer. Under single coverage, `COMEBin` achieved the highest Total Score, but its Clean Bin Bases were lower than those of several marker free methods. `QuickBin` and `QuickBin` [UL] retained broad genome representation and high sequence recovery while preserving stronger clean bin recovery than the highest yield SCG marker-based methods (Fig. 3g). Under multi coverage, `QuickBin` [UL] achieved the highest Total Score, while `QuickBin` produced the highest Clean Bin Bases with complete genome representation (Fig. 3h). This pattern supports the intended distinction between `QuickBin` modes, with `QuickBin` [UL] favoring additional recovery and default `QuickBin` retaining the strongest purity signal.

The controlled synthetic benchmarks show that `QuickBin` balances genome recovery with bin purity across community sizes and coverage

designs. CheckM2 based ranking emphasized estimated MAG yield, whereas contig origin scoring resolved differences in sequence recovery and Clean Bin Bases. The clearest QuickBin advantage appeared in the large multi coverage setting, where multiple coverage profiles helped recover a broad genome set while maintaining high clean bin recovery (Fig. 3h). These results support the use of controlled synthetic ground truth as a complement to CAMI II when evaluating high fidelity metagenome binning.

Real metagenome benchmarks show robust recovery of high-fidelity MAGs by QuickBin

The CAMI II and controlled synthetic benchmarks showed that estimated MAG yield and clean bin recovery can rank binners differently. We next tested whether QuickBin's purity oriented design remained useful on real metagenomes, where contig origin truth is not available, and Clean Bin Bases cannot be measured. We therefore evaluated three practical endpoints available for real assemblies: run completion, CheckM2 estimated recovery of prokaryotic MAGs, and EukCC estimated recovery of eukaryotic MAGs. The benchmark included 297 public JGI metagenomes spanning Aquatic, Engineered, Plants and Terrestrial biomes (Fig. 4a), with Illumina and PacBio assemblies covering a broad range of assembly sizes (Fig. 4b, c). Here, high fidelity refers to the strict quality tier used throughout this study, at least 95% completeness and at most 1% contamination.

Run completion differentiated binners that were robust at scale from binners with higher failure rates on real metagenomes. QuickBin, QuickBin [UL], VAMB and MetaBAT2 completed binning in all metagenomes, whereas COMEBin and both SemiBin2 modes had incomplete runs under the same wall time and memory limits (Fig. 4d). Completion status was included as a benchmark endpoint because incomplete runs produce no bins for subsequent quality filtering in automated analyses. QuickBin completed all binning tasks without GPU hardware, CheckM1 based preselection, or SCG marker-based binning (Fig. 4d).

On Illumina metagenomes, QuickBin [UL] recovered the largest set of CheckM2 estimated high fidelity prokaryotic MAGs, and default QuickBin was also among the strongest methods (Fig. 4e). The ranking matches the mode behavior observed in controlled synthetic benchmarks, where QuickBin [UL] favored recovery and default QuickBin maintained the more conservative setting. Because real metagenomes lack contig origin truth, these counts represent estimated high fidelity MAG yield rather than direct clean base purity.

On PacBio metagenomes, QuickBin did not lead prokaryotic MAG yield. SemiBin2, SemiBin2 [SS], MetaBAT2, COMEBin and VAMB recovered more strict tier prokaryotic MAGs than QuickBin (Fig. 4f). These results define a platform limit for the current implementation, which was designed and optimized primarily for short read metagenome workflows. QuickBin nevertheless recovered strict tier prokaryotic MAGs from PacBio assemblies, indicating usable but lower yield on long read assemblies.

Eukaryotic MAG (eukMAG) recovery showed a different ranking from prokaryotic MAG yield. In Illumina metagenomes, QuickBin, QuickBin [UL] and MetaBAT2 recovered the broadest eukMAG sets across EukCC completeness bins, whereas COMEBin and both SemiBin2 modes produced fewer eukMAGs (Fig. 4g). The recovery of eukMAGs by QuickBin is consistent with its marker free design, which does not restrict binning to bacterial or archaeal SCG signal. In PacBio metagenomes, eukMAG counts were low for all methods. QuickBin and QuickBin [UL] recovered eukMAGs in this platform subset, but the high completeness eukMAG signal was stronger for VAMB and SemiBin2 (Fig. 4h).

Biome stratification showed that the Illumina result was distributed across environmental categories. At CheckM2 estimated contamination at most 1%, QuickBin and QuickBin [UL] produced MAGs across completeness bins in Aquatic, Engineered, Plants and Terrestrial metagenomes, with consistent representation in the at least 95% completeness tier (Supplementary Fig. 3a). In PacBio metagenomes, binner ranking varied by biome and QuickBin recovered fewer low contamination MAGs than the leading methods in several panels (Supplementary Fig. 3b). This stratified analysis supports the same platform distinction observed in Fig. 4e, f, with stronger QuickBin recovery on Illumina assemblies than on PacBio assemblies.

Together, the real metagenome benchmarks extend the ground truth analyses to datasets where purity can only be inferred indirectly. QuickBin combined full run completion, the highest estimated high fidelity prokaryotic MAG yield on Illumina metagenomes through QuickBin [UL], and broad eukMAG recovery under EukCC. The same analysis also identified reduced prokaryotic MAG yield on PacBio assemblies. These results support QuickBin as a practical binner for large short read metagenome studies that prioritize high confidence genome recovery over maximal bin count alone.

Extrachromosomal element benchmarks show that QuickBin captures host plasmid signal conservatively

The previous benchmarks focused on chromosomal MAG recovery, where binner performance can be evaluated by estimated completeness, estimated contamination, and, in synthetic settings, contig origin truth. We next examined a more difficult special case for genome-resolved metagenomics. Extrachromosomal elements, including plasmids, are challenging for binning because their copy number, coverage profile, and sequence composition can differ from their host chromosome. These properties affect the main signals used by most binners, including differential coverage and sequence composition. For a purity-oriented binner, the expected behavior is therefore not aggressive host plasmid merging, but assignment only when the available evidence supports a shared origin. We evaluated whether each binner assigned host and plasmid sequence to one bin, recovered host and plasmid fragments across multiple bins, or failed to assign host and plasmid together. The benchmark included three data configurations, ONT hybrid reference with Illumina reads, PacBio hybrid reference with PacBio reads, and Illumina assembly with Illumina reads (Supplementary Fig. 4). Across configurations, complete assignment of host and plasmid to one bin was uncommon for all binners, confirming that extrachromosomal elements are not recovered as a simple extension of chromosomal MAG binning.

In the ONT hybrid reference evaluated with Illumina reads, most binners did not assign host and plasmid sequence together. QuickBin and QuickBin [UL] were the only methods that recovered host plasmid signal in this configuration, including a small number of same bin assignments and additional split assignments across multiple bins (Supplementary Fig. 4). In the PacBio hybrid reference evaluated with PacBio reads, no method consistently placed host and plasmid in one bin. QuickBin and QuickBin [UL] recovered the broadest split assignment signal, indicating recovery of host and plasmid sequence without full consolidation into one MAG (Supplementary Fig. 4).

The Illumina benchmark produced the strongest overall host plasmid signal. QuickBin and QuickBin [UL] recovered most cases as split assignments across multiple bins, while MetaBAT2 and both SemiBin2 modes recovered fewer linked cases and VAMB and COMEBin largely failed to assign host and plasmid together (Supplementary Fig. 4). Same bin assignments remained rare, supporting the interpretation that full host plasmid consolidation requires unusually strong evidence under standard binning features.

The extrachromosomal element benchmark identifies a boundary condition for all evaluated binners. Plasmids and related elements are difficult to place because the biological features that distinguish them from host chromosomes overlap with the same signals used to prevent chimeric bins. QuickBin recovered host plasmid signal in several settings, but generally avoided complete host plasmid merging. This pattern is consistent with the central goal of QuickBin, which is to preserve bin purity rather than maximize bin size when origin evidence is ambiguous.

QuickBin computational efficiency supports deployable high fidelity binning at scale

High fidelity binning is most useful when it can be applied across large metagenome collections under realistic resource limits. The preceding

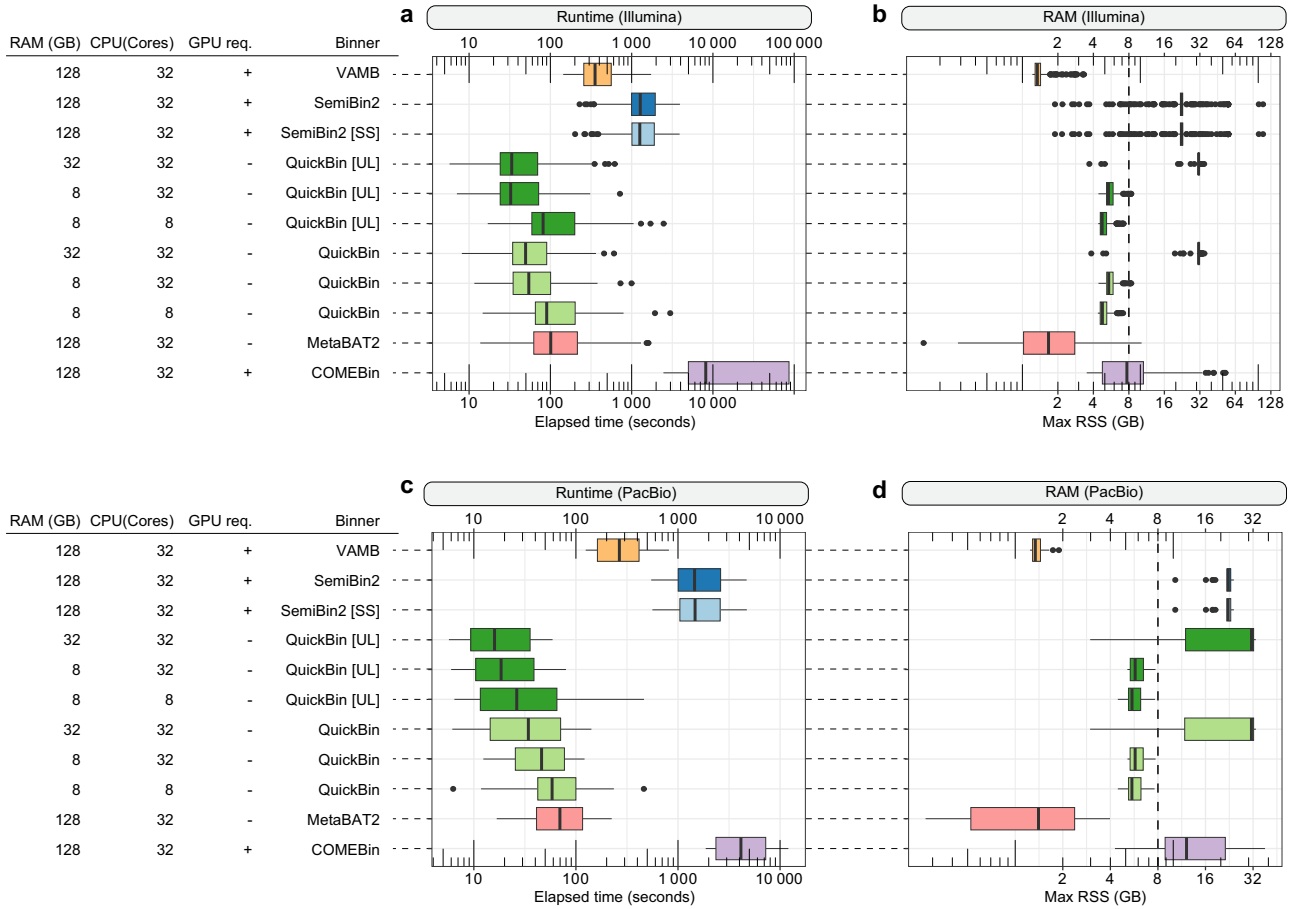


Fig. 5 | Runtime and peak memory usage across binners highlight QuickBin efficiency and deployability. Per metagenome elapsed time and peak memory usage for each binner and resource configuration. Panels show runtime for Illumina assemblies in (**a**) Max RSS for Illumina assemblies in (**b**) runtime for PacBio assemblies in (**c**) and Max RSS for PacBio assemblies in (**d**). The left labels indicate RAM allocation, CPU cores, GPU use, and binner. `QuickBin` and `QuickBin` [UL] were evaluated under CPU only configurations with 8 GB RAM and 8 or 32 CPU cores, and with 32 GB RAM and 32 CPU cores. Other binners were evaluated under the resource allocations shown. +GPU indicates GPU enabled runs and -GPU indicates CPU only runs. Max RSS denotes maximum resident set size, the peak physical memory used during execution. `QuickBin` [UL] indicates ultraloose mode, and `SemiBin2` [SS] indicates self supervised mode. For **a**, **b**, $n = 275$ biologically independent Illumina metagenome samples; for **c**, **d**, $n = 22$ biologically independent PacBio metagenome samples.

benchmarks evaluated recovery and purity. We next quantified whether the same design choices also produced a computational profile compatible with routine deployment. We profiled elapsed time and peak memory usage across Illumina and PacBio real metagenomes under standardized resource caps, and compared `QuickBin` with `VAMB`, `SemiBin2`, `SemiBin2` [SS], `MetaBAT2` and `COMEBin` (Fig. 5 and Supplementary Fig. 5).

`QuickBin` and `QuickBin` [UL] were evaluated in CPU only configurations spanning 8 GB RAM with 8 CPU cores, 8 GB RAM with 32 CPU cores, and 32 GB RAM with 32 CPU cores. Other binners were evaluated under the resource allocations shown in Fig. 5, including GPU enabled configurations where supported. For `QuickBin`, peak Max RSS (maximum resident set size) should be interpreted relative to the configured Java heap. The 32 GB runs therefore reflect a larger permitted heap, whereas the 8 GB runs define the constrained memory configuration.

On Illumina metagenomes, `QuickBin` configurations occupied the shortest runtime range among the evaluated methods, together with `MetaBAT2`, while `VAMB` and the `SemiBin2` modes required longer runtimes, and `COMEBin` had the longest runtime distribution (Fig. 5a). `QuickBin` also maintained low peak memory in the 8 GB configurations, whereas higher resource configurations and other binners showed broader or higher Max RSS distributions (Fig. 5b). These patterns indicate that `QuickBin` can run within constrained CPU only settings without the GPU or high memory allocations used by several alternative methods.

The PacBio benchmark showed the same computational pattern. `QuickBin` configurations remained in the shortest runtime range, whereas `COMEBin` and the `SemiBin2` modes required longer runtimes (Fig. 5c). In memory, the 8 GB `QuickBin` configurations remained close to their constrained allocation, while 32 GB configurations used more memory when permitted, especially for `QuickBin` [UL] (Fig. 5d). This behavior is consistent with a tunable resource profile rather than a fixed high memory requirement.

Scaling analyses against assembly size supported these observations. Across Illumina and PacBio assemblies, `QuickBin` showed shallow runtime trends across the tested size range, while `COMEBin` increased more strongly with assembly size (Supplementary Fig. 5a, c). In memory scaling, the constrained `QuickBin` configurations remained near the 8 GB reference line, whereas several other methods increased with assembly size or approached higher memory bands (Supplementary Fig. 5b, d).

We further tested `QuickBin` on the large coassembly from the Terabase Scale Coassembly of a Tropical Soil Microbiome project[44]. The input comprised a 74 GB contig file with 55,342,847 contigs and 95 sorted read alignment BAM files totaling 1.4 TB. `QuickBin` completed this analysis in 10 h 56 min 23 s using 64 threads, with peak memory usage of 383 GB, and produced 5028 bins. These results show that `QuickBin` combines high-fidelity binning with low runtime and configurable memory usage. The efficiency profile follows from the same architecture used to preserve bin

purity, where spatial indexing and early comparison filters limit unnecessary contig pair evaluation. This supports QuickBin as a practical CPU only option for large metagenome studies where high confidence genome recovery must be coupled to reproducible execution under bounded computational resources.

Discussion

Genome resolved metagenomics continues to expand in scale and impact, but a persistent methodological challenge remains. Methods that maximize MAG yield do not necessarily maximize MAG fidelity, and methods with strong recovery can be difficult to deploy broadly because of compute and hardware requirements[19,22]. This tradeoff matters because comparative genomics, ecological inference, and curated genome resources increasingly depend on consistently high quality reconstructions rather than maximal genome counts alone[1,17,45]. In this study, we show that QuickBin addresses this gap by prioritizing near-complete, ultra-low-contamination MAG recovery while remaining deployable under bounded computational resources.

QuickBin is intended for genome-resolved analyses that require stringent bin quality across many assemblies, especially the target used throughout this study, at least 95% completeness and at most 1% contamination. Such projects often include dozens to thousands of single-coverage or multi-coverage assemblies, where compute limits and incomplete runs can constrain binner choice and quality filtering. By combining conservative merging with spatial candidate restriction and early comparison filters, QuickBin shifts the workflow toward fewer but higher confidence genome bins for comparative genomics and ecological inference (Fig. 1).

CAMI II provided an independent benchmark because it uses complex communities with known contig origin truth and has become a reference framework for metagenomic software evaluation[19]. In our analyses, CAMI II did not identify a universal leading binner. CheckM2 estimates favored SCG marker-based methods in several panels, whereas contig origin scoring separated number of MAGs recovered from bin purity and showed competitive clean bin recovery by QuickBin in marine, plant, and toy mouse gut communities (Fig. 2). This result is consistent with recent benchmarking work showing that binner rankings depend on dataset structure, sequencing strategy, and evaluation endpoint[22,43]. It also motivates the controlled synthetic benchmarks used here, where community size and coverage design could be varied.

Across real Illumina metagenomes, QuickBin [UL] produced the largest CheckM2 estimated high-fidelity prokaryotic MAG set at the strict tier of at least 95% completeness and at most 1% contamination, while default QuickBin was also among the strongest methods (Fig. 4e). Under the same resource caps, QuickBin completed all real metagenome runs (Fig. 4d) and had short CPU-only runtimes under constrained memory configurations (Fig. 5a, b). COMEBin and both SemiBin2 modes had incomplete runs under the same resource caps, consistent with recent large-scale benchmarks showing that recovery, runtime, memory, and hardware availability should be considered together when comparing binners[22,43]. These results position QuickBin as a practical alternative when the goal is to recover a high-confidence genome set rather than maximize estimated bin count alone. Given that QuickBin does not use SCG information during binning and is CheckM agnostic, its output is not selected by the same marker signals used by downstream quality estimators.

The controlled synthetic benchmarks refine this interpretation by separating estimated MAG yield from ground truth bin purity. CheckM2 based ranking emphasized estimated MAG counts, whereas contig origin scoring exposed differences in Clean Bin Bases that were not captured by estimated completeness and contamination alone (Fig. 3). This distinction is central because high sequence recovery can coexist with lower bin purity. Marker based quality tools remain essential for real metagenomes, but direct ground truth mapping is needed when the goal is to evaluate clean base recovery rather than estimated MAG count alone[19,37]. Under this condition, QuickBin narrowed the gap between classic coverage and composition binners and more compute intensive modern approaches in the high fidelity regime, while retaining marker free applicability across prokaryotic and eukaryotic fractions[1,11,13,15,36,37]. Consistent with this marker free design, QuickBin also recovered broad eukMAG sets under EukCC, especially in Illumina metagenomes (Fig. 4g, h); recovery of eukMAGs is an area of growing interest given the emerging role of eukaryotic MAG catalogs in ecology and evolution[46].

As expected from its design focus on short read workflows, QuickBin did not lead prokaryotic MAG yield on PacBio assemblies (Fig. 4f). Nevertheless, QuickBin recovered strict tier PacBio MAGs and eukMAGs across multiple EukCC completeness bins (Fig. 4f–h). This result defines a platform boundary for the current implementation rather than a limitation of the high fidelity objective. Long read metagenomic assembly and binning are progressing rapidly, with improved HiFi assembly, long read strain resolution and long read specific binners changing expected performance on these datasets[47–50]. Future QuickBin development will include optimization for long read assemblies.

The extrachromosomal element benchmark adds a related boundary condition. Plasmids can differ from host chromosomes in copy number, coverage and sequence composition, and recent plasmid reconstruction methods explicitly model these differences through assembly graph, alignment graph, coverage and composition signals[51–53]. QuickBin recovered host plasmid signal in several configurations, but same bin host plasmid assignment remained rare, consistent with a purity focused design in which ambiguous mobile element links are not forced into chromosomal MAGs (Supplementary Fig. 4).

A central practical outcome of this study is that QuickBin's high fidelity performance is paired with low runtime and configurable memory use (Fig. 5; Supplementary Fig. 5). QuickBin ran in CPU only configurations, including 8 GB RAM settings, while several alternative methods were evaluated with GPU enabled or higher memory configurations. These characteristics address deployability bottlenecks emphasized in recent community assessments, where binner performance must be considered together with throughput, hardware accessibility and failure mode[19,22,43]. This combination supports QuickBin as a high confidence binner for large short read studies and curated genome collections compatible with GTDB based taxonomy and modern genome resource pipelines[17,45,54,55].

Our results also have implications beyond traditional comparative genomics. Large MAG collections are increasingly reused as training data for nucleotide and protein language models, including nucleotide and protein language models like Nucleotide Transformer[23], Evo[25], and ESM-class models[27]. In this context, bin purity is a data quality issue because chimeric or mislabeled sequence can alter the signals learned by downstream models. By recovering high confidence genome sets with at least 95% completeness and at most 1% contamination, QuickBin supports genome resolved metagenomic resources that are better aligned with both biological inference and data centric modeling.

Current MAG standards and common benchmark summaries still rely heavily on marker based completeness and contamination estimates. These tools are essential for real metagenomes, but they have limited resolution for distinguishing very low contamination from no detectable contamination and may underrepresent contamination carried by short or marker poor contigs[1]. Our CAMI II and controlled synthetic analyses show that estimated MAG yield, Total Score, genome representation, sequence recovery, and Clean Bin Bases can rank binners differently (Figs. 2 and 3). Ground truth benchmarks therefore remain necessary for auditing estimator behavior and for defining targets beyond conventional high quality MAG thresholds. A high stringency category such as at least 95% completeness and at most 1% contamination would better reflect applications where MAGs are used as isolate-like references or as training data for genome scale models.

In summary, QuickBin provides a deployable path to high fidelity genome reconstruction at scale. Across CAMI II, controlled synthetic and real metagenome benchmarks, QuickBin showed strongest performance when the endpoint emphasized clean bin recovery, strict Illumina MAG

yield and CPU only deployment. The same analyses also defined clear limits, including lower prokaryotic MAG yield on PacBio assemblies and conservative host plasmid consolidation. QuickBin's spatial indexing and early comparison filters support short runtimes and bounded memory use by avoiding global all to all contig comparisons. These properties make QuickBin well suited for large scale, high confidence genome catalog construction and for workflows where MAG fidelity, reproducibility, and compute accessibility are primary optimization objectives.

Methods

QuickBin's algorithm - pseudocode

We present below the pseudocode for Phases 3–5 and the Oracle comparison cascade, to supplement the existing descriptions presented in the Results section.

Algorithm 1. Oracle comparison cascade

The Oracle evaluates a candidate bin pair (a, b) through four stages of increasing computational cost. Each stage can reject the pair via early exit, avoiding unnecessary computation.

```
function ORACLE_COMPARE(a, b, thresholds, network):
// Stage 1: Fast scalar comparisons (~70% rejected here)
gcDif ← |a.gc - b.gc|
hhDif ← |a.hh - b.hh| × 1.65
cagaDif ← |a.caga - b.caga| × 1.55
gchhDif ← max(gcDif, hhDif, cagaDif)
depthRatio ← maxSampleDepthRatio(a, b)
  if gchhDif > maxGCDif or depthRatio > maxDepthRatio:
    return REJECT
  // Edge weight bonus: relax thresholds for well-connected pairs
edgeMult ← computeEdgeMult(a, b)
if gchhDif > maxGCDif × edgeMult or depthRatio > maxDepthRatio ×
edgeMult or covariance(a, b) > maxCovariance × edgeMult:
return REJECT
// Stage 2: Mid — trimer comparison (~94% of survivors rejected)
trimerDif ← cosineDifference(a.trimers, b.trimers) // SIMD-accelerated
if trimerDif > max3merDif × edgeMult:
  return REJECT
// Stage 3: Slow — tetramer/pentamer (~94% of survivors rejected)
tetramerDif ← cosineDifference(a.tetramers, b.tetramers)
pentamerDif ← cosineDifference(a.pentamers, b.pentamers)
kmerProb ← empiricalProbability(min(|a|, |b|), tetramerDif)
product ← tetramerDif × depthRatio
if tetramerDif > max4merDif × edgeMult or product > maxProduct ×
edgeMult or pentamerDif > max5merDif × edgeMult or kmerProb <
minKmerProb:
return REJECT
// Stage 4: Neural network (applied to surviving ~0.1%)
similarity ← heuristicScore(depthRatio, gcDif, tetramerDif, covariance,
kmerProb)
if network ≠ null:
  vector ← buildFeatureVector(a, b, 28 features) // see Table 1
  netOutput ← network.feedForward(vector)
  adjust edgeMult based on netOutput vs. cutoff
  re-check all thresholds with adjusted edgeMult
  if any threshold exceeded:
    return REJECT
  return netOutput
return similarity
```

Note: All thresholds are scaled by sizeAdjustMult(min(|a|, |b|)), which relaxes cutoffs for smaller contigs and tightens them for larger ones.

Algorithm 2:. Phase 3 — BinMap Construction (Greedy Within-Cell Merging)

```
function MAKE_BINMAP(contigs):
  sort contigs by size descending
  binMap ← empty spatial grid (GC quantized at Δ=0.02, depth at 2×
  log scale)
  residual ← empty list
for each contig c in contigs:
  if|c| < 2500:
    residual.append(c)
    continue
cell ← binMap.quantize(c.gc, c.depth)
merged ← false
for each existing bin b in cell:
  if |b| ≥ minSizeToCompare:
    score ← ORACLE_COMPARE(c, b, strict_thresholds)
    if score > 0:
      b.merge(c) // add c's k-mers, depth, GC to b
      merged ← true
      break    // greedy: take first match
  if not merged:
  binMap.add(newCluster(c), cell)
return binMap, residual
```

Strict thresholds for Phase 3: maxKmerDif=0.002, maxGCDif=0.015, maxDepthRatio=1.05, maxCovariance=0.0001. Because contigs are processed in size-descending order, large contigs anchor clusters before smaller, noisier contigs are considered.

Algorithm 3:. Phase 4 — Cluster refinement

```
function REFINE_BINMAP(binMap):
  totalMerged ← 0
for each (phase, stringency, range, passes) in refinement_schedule:
  // Default schedule (without sketch):
  // AA: stringency = 0.8, range=baseRange + 1, large minSize
  // A: stringency = 0.8, range=baseRange
  // D: stringency = 0.9, range=baseRange
  // E: stringency = 1.0, range=baseRange + 2
thresholds ← base_thresholds × stringency
for pass = 1 to passes:
  clusters ← sorted(binMap.clusters, by size descending)
  mergeTargets ← empty map
// Multithreaded comparison
parallel for each cluster a in clusters:
  if|a|< minSize × pass:
    continue
best ← binMap.findBestCluster(a, range, thresholds)
  // searches (1 + 2r)² neighbor cells in the grid,
  // e.g., range = 1 → 9 cells, range = 3 → 49 cells
if best ≠ null and best ≠ a:
  mergeTargets[a] ← best
// Single-threaded merge (prevents concurrent modification)
for each (a → best) in mergeTargets, smallest a first:
  best.merge(a)
rebuild binMap from surviving clusters
totalMerged += |mergeTargets|
if |mergeTargets| < 2: break // converged
return totalMerged
```

Stringency scales all Oracle thresholds: higher stringency = more permissive merging. The schedule progresses from strict (0.8×) to permissive (1.0×), and from narrow search radius to wide, capturing increasingly distant associations.

Algorithm 4:. Phase 5 — Edge Following

```
function FOLLOW_EDGES(contigs, binMap):
totalMerged ← 0
for round = 1 to 5:
  bins ← all clusters and unclustered contigs
// Multithreaded: find merge targets via edge connectivity
parallel for each bin a in bins:
```

```
    edges ← a.pairMap sorted by weight descending
    bestScore ← 0
    target ← null
  maxEdges ← 4 if a is Contig else min(16, 4 + 2×(numContigs-1))
  minWeight ← max(minEdgeWeight, 0.4 × edges[0].weight)
  for i = 0 to min(maxEdges, |edges|):
    if edges[i].weight < minWeight: break
    c ← contigs[edges[i].key]
    b ← c.cluster or c // resolve to cluster
    if a = b: continue
  // Require reciprocal edge
  reciprocal ← min(c.edgesTo(a), edges[i].weight)
  if reciprocal ≥ minWeight:
    score ← ORACLE_COMPARE(a, b, edge_thresholds)
    if score > bestScore:
      target ← b
      bestScore ← score
  a.dest ← target.id if target ≠ null
  // Single-threaded merge
  merged ← mergeAllWithDest(contigs, bins)
    totalMerged += merged
    if merged ≤ 1: break // converged
  return totalMerged
```

Edge-following is O(N) expected because each bin examines at most a constant number of edges (4 for contigs, 16 for clusters). The comparison methodology is identical to refinement, as only the pair selection mechanism differs (graph edges vs. spatial proximity).

Details on `QuickBin`'s neural network architecture and training

Architecture. The neural network is a feed-forward network with 7 layers (including input and output): 28 input dimensions, five hidden layers of 128, 64, 112, 48, and 12 nodes respectively, and a single output node, for a total of approximately 19,000 edges and 364 nodes. The network uses structural sparsity: at initialization, edges are randomly removed to achieve 75% connectivity (density = 0.75). No dropout is used during training or inference; the sparse topology is established once at initialization and remains fixed throughout. The sparsity pattern is block-aligned, and minimum connectivity per node is enforced to prevent dead neurons.

Activation functions are assigned randomly per node at initialization from a weighted distribution with unnormalized rates of 60% sigmoid, 40% tanh, 2% mirrored sigmoid (MSig), and 2% rslog (normalized to sum to 1.0 before assignment). This per-node diversity means that within a single hidden layer, nodes apply different activation functions, producing an implicit ensemble-like effect. The four activation functions used are:

- Sigmoid: $\sigma(x) = \frac{1}{1/(1+e^{-x})}$, output range (0, 1). The standard logistic function.
- Tanh: $\tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$, output range (−1, 1). Zero-centered, which can accelerate convergence.
- MSig (mirrored sigmoid): a bell-shaped function constructed from two offset sigmoids that peaks at $x = 0$ and decays symmetrically, output range (0, ~1). Useful for detecting "similarity" where values near zero indicate a match.
- RSLog (rotationally symmetric logarithm): $rslog(x) = sign(x) \cdot \log(|x| + 1)$, unbounded output range. Its derivative is $\frac{1}{(|x|+1)}$, which approaches zero slowly rather than exponentially, eliminating the vanishing gradient problem that causes sigmoid and tanh to saturate.

The final output layer uses *rslog* exclusively to ensure that error gradients propagate effectively back through the network regardless of output magnitude, preventing the gradient saturation that would occur with a sigmoid or tanh output node. In the hidden layers, the random mixture of functions with different saturation behaviors, output ranges, and gradient profiles creates functional diversity that acts as an implicit ensemble within the network.

Training procedure. The network was trained using BBTools' Trainer (invoked via `train.sh`), which implements a batch-based stochastic training procedure rather than epoch-based iteration in the conventional sense.

The training data (~13.8 million unique vectors from 6 input files) is randomly split 90/10 into exclusive training and validation sets using a fixed random seed. The training set (~12.4 million examples) is further partitioned into approximately 207 subsets of ~60,000 samples each. The active subset rotates every 64 batches; when all subsets have been visited, the data is reshuffled, and subsets are rebuilt. On each training step, 8% of the current subset is sampled (approximately 5000 samples), and weights are updated via backpropagation. No single pass over the full dataset is performed, and not every training example is guaranteed to be used in a given rotation.

The learning rate starts at 0.03 and ramps up by a factor of 5 over the first 640 batches. After the peak, it decays exponentially over the remaining training, with a secondary decay factor of 2.8 applied over the total training duration.

Training proceeds for a fixed number of 800,000 batch updates, with approximately 4.3 billion cumulative sample-passes (corresponding to roughly 350 effective passes over the full training set on average), though priority-based sampling and mid-training data reduction (described below) cause substantially unequal coverage across examples. Five independent networks are trained in parallel from different random initializations. Throughout training, each network is periodically evaluated on the held-out validation set, and the best-performing state (by false positive rate, with fallback to weighted error rate) is checkpointed. The final model is the best checkpoint across all five networks and all batch steps. This is functionally analogous to early stopping with checkpoint selection—if the optimal state occurs at batch 270,000, subsequent training does not affect the final result—but without explicit validation-based halting. The approach trades wall-clock time for result quality, as training continues past the optimum to ensure the optimum has been found.

The classification threshold is determined via crossover analysis on the validation set: the cutoff is placed where the false positive rate crosses the false negative rate, balancing both error types equally.

Several mechanisms manage training stability:

- Weight normalization: bias and weight values are z-score normalized per layer, blended with the original values at a decaying rate (initial factor 0.125, shrink rate 0.999 per update).
- Simulated annealing: random noise is injected into weights with initial strength 0.003 and 22.5% probability per eligible batch. Annealing strength decays exponentially and is disabled after 80% of training.
- Amplitude dampening: weight updates that would increase a weight's magnitude beyond 0.1 are multiplied by 0.98, preventing runaway growth.
- Sample triage: after 20% of training, the top 0.1% most extreme positive and negative examples are temporarily deferred (for 500 batches) to reduce sensitivity to outliers.
- Sample dumping: at 25% of training, the 50% lowest-error samples are removed from the training set (retained for validation), focusing subsequent training on hard cases near the decision boundary. After dumping, the sampling fraction is rescaled to compensate for the reduced dataset.
- Weight ramp-up: during the first 3% of training, weight updates are scaled by $\sqrt{\frac{batch}{ramp_length}}$, preventing large early updates from destabilizing a randomly initialized network.
- Error emphasis: false positive and false negative errors receive 9× emphasis relative to correct classifications, implementing cost-sensitive learning. Conversely, over-confident correct predictions (predictions that overshoot the target in the correct direction) are down-weighted by 0.2×, discouraging the network from wasting capacity on already-correct classifications.

- Class balancing: the underrepresented class is oversampled (by cloning existing examples) to 40% of the majority class size, reducing class imbalance bias.
- Decision boundary enforcement (spread): a buffer zone extending 0.1 on each side of the classification cutoff defines a zone where predictions receive the full false-positive or false-negative error penalty even if they are technically on the correct side. This forces the network to make confident classifications rather than clustering predictions near the decision boundary, where small perturbations could flip the outcome.

Weights are drawn from a three-component mixture: 30% uniform (±0.25), 40% exponential ($\lambda = 5$), and 30% cubic, clamped to ±2.0. Biases are drawn from a separate distribution: 80% uniform (±0.25) and 20% cubic, with non-zero values guaranteed. Biases are not clamped.

The final model achieves a false positive rate of 1.9% and a false negative rate of 1.9% on the validation set. Training required approximately 7 h on 30 CPU cores with 87 GB RAM.

Training data. Training vectors were generated from three sources to ensure the network generalizes across a range of metagenomic characteristics:

- Synthetic metagenomes: contigs were assembled by `Tadpole` (https://bbmap.org/tools/tadpole) from paired-end reads generated by `RandomReadsMG`, using reference genomes at multiple depth distributions (uniform, exponential, root, and minimum-4) across up to 9 simulated sequencing samples. Coverage files were generated by mapping reads with `BBMap`[56] and computing per-contig depth via `QuickBin`'s validate mode. Ground-truth labels were assigned based on the dominant contributing reference organism per contig, as described in section "Workflow for synthetic data generation".
- CAMI benchmark data: vectors from the Critical Assessment of Metagenome Interpretation (CAMI) community benchmark datasets were included to expose the network to independently generated metagenomic assemblies with known ground truth. The CAMI datasets used for neural-network training were disjoint from the six CAMI II panels used for evaluation.
- Real hard cases: During actual `QuickBin` binning runs on the synthetic data, false positive merges (incorrect merges accepted by the heuristic cascade) and false negative rejections (correct merges rejected by the cascade) were emitted as additional training vectors. This feedback loop captures the specific pair types that the cascade struggles with, focusing training on the decision boundary that matters in practice.

Vectors were generated using `makequickbinvector.sh` (AllToAllVectorMaker), which selects random pairs of contigs across multiple length bins (500 bp to >800 kbp) and computes the 28-dimensional feature vector for each pair. Both restricted sampling (pairs filtered to narrow GC, k-mer, and depth ratio ranges, focusing on ambiguous cases near the decision boundary that could actually bypass early exits and be evaluated by the neural network) and unrestricted sampling (no pairwise filters, covering the full feature space) were used. All vectors were deduplicated and combined, yielding approximately 13.8 million unique training vectors.

Evaluation. The training data is randomly split 90/10 into exclusive training and validation sets at load time using a fixed random seed for reproducibility. The validation set is used during training for checkpoint selection (best model by false positive rate at the FNR = FPR crossover point) according to the validation set.

Software used for benchmarking

Details on version, installation method, and running methods for software and databases used in this project can be found in Supplementary Table S1. In brief, software used in this study and corresponding versions were as follows: Binning was performed with `QuickBin` (BBTools 39.60; deployed via `docker` image `bryce911/bbtools:39.60`), `MetaBAT2` v2.18, `SemiBin2` v2.2.0, `COMEBin` v1.0.4, and `VAMB` v5.0.4; all binners and their respective dependencies, including `CUDA` libraries required for GPU usage, were installed via `pixi` unless otherwise noted. Quality assessment and filtering used `CheckM2` v1.1.0 (installed via `pixi`) and `EukCC` v2.1.3 (executed from the container `quay.io/microbiome-informatics/eukcc:2.1.3`), and `gradebins` from `BBTools` (v39.54; deployed via the `Docker` image at `bryce911/bbtools:39.54`). Taxonomic assignment used `GTDB-Tk` v2.4.1 (installed via `mamba` environment). Read mapping processing used `samtools` v1.22.1 (installed via `pixi`). Reference databases included the `GTDB-Tk` data package for GTDB release r226 and the `EukCC` database v1.2, both retrieved via `wget`.

Metagenome datasets, assemblies, alignments, and reference databases

Metagenomes used for benchmarking ($n = 297$) were obtained from IMG/M[57] through the JGI Data Portal (data.jgi.doe.gov), including assembled contigs and the corresponding read-alignment files (SAM/BAM) used for coverage profiling. Sample identifiers, project identifiers, sequencing platform labels (Illumina or PacBio), and biome assignments used throughout the manuscript are provided in Supplementary Table S2, enabling retrieval of the exact input datasets used for all comparisons. Biomes were selected and grouped into the four categories: Aquatic, Engineered, Plants, and Terrestrial (Supplementary Table S2). For real metagenomes, assemblies spanned a broad range of total assembled sequence, from 101,945,291 bp to 3,233,917,651 bp. Illumina metagenomes selected were only those assembled with `metaSPAdes`[39], and PacBio metagenomes selected were only those assembled with `metaFlye`[58] (sequencing platform detailed per sample in Supplementary Table S2). All binners were run on the same assemblies and associated coverage inputs derived from these contigs and alignments, and downstream quality-control evaluations were applied uniformly across methods.

We selected real metagenomes using IMG metadata to construct a recent, public, and ecologically annotated benchmarking set suitable for consistent comparisons. Starting from restricting to metagenome analysis projects added on or after 1 January 2019 and required that sequencing was exclusively done by the DOE Joint Genome Institute as we know the thorough QA/QC these projects undergo. All datasets were unrestricted and of public availability. To standardize inputs, we limited assemblies to those generated with `metaSPAdes` or `metaFlye` and restricted sequencing platforms to Illumina or PacBio. We retained only samples with geographic coordinates (latitude and longitude) to supplement metadata with biome- and geography-aware information. To avoid atypical assemblies that complicate scaling comparisons, we constrained the total assembled sequence to 100 Mbp to 10 Gbp. Finally, we required non-unclassified ecosystem annotations to enable robust stratification into the biome and sub-biome categories used throughout the benchmarking.

Reference databases were version-pinned. For prokaryotic genome quality estimation, we used the `CheckM2` reference database v1.1.0. For eukaryotic genome quality estimation, we used the `EukCC` database v1.2. For prokaryotic taxonomy, we used the `GTDB-Tk` data package corresponding to GTDB release r226.

Workflow for synthetic data generation

To ensure rigorous training and validation of the neural network, we developed a data generation methodology designed to mimic the structural and topological artifacts of real metagenomic assemblies (e.g., breakpoints, collapsed repeats, and uneven coverage) rather than utilizing idealized "shredded" references that oversimplify the features of metagenomes. The pipeline follows these steps:

Topological re-assembly. Reference genomes were individually re-assembled using `Tadpole` with an extended k-mer length ($k = 155$) and permissive extension parameters (*mincountseed = 1, mincountextend = 1*). This step forces the assembler to traverse the de Bruijn graph for

single-copy regions while naturally collapsing repetitive elements, thereby producing contigs with realistic graph topology and coverage spikes, rather than uniform linear fragments.

Synthetic read generation. Error-free synthetic paired-end reads[42] were generated from the original reference genomes using RandomReads at 10× depth. Taxonomic identifiers (TaxIDs) were encoded into the source filenames to maintain provenance.

Strict alignment and annotation. Synthetic reads were mapped back to the re-assembled contigs using `BBMap`[56] in "semiperfect" mode (`semiperfectmode`, `ambig=random`). This alignment mode enforces perfect identity for the mapped portion of the read but permits soft-clipping at contig termini, ensuring that valid alignments are retained at breakpoints and contig ends where assemblers typically truncate.

Ground-truth assignment. Contigs were annotated using ContigRenamer, a custom utility (`renamebymapping.sh`: https://bbmap.org/tools/renamebymapping) that parses alignment headers to calculate per-contig coverage and assigns the taxonomic ID via a majority-vote consensus of mapped reads. This ensures that chimeric contigs or collapsed repeats are labeled based on the dominant contributing organism, providing a realistic "ambiguous" class for the neural network to evaluate. The inputs to the network are detailed in Table 1.

Controlled synthetic (ground-truth) metagenome reads

Controlled synthetic metagenomes were generated to evaluate binner behavior under known contig origin truth while varying community size and coverage design. Two communities were used for benchmarking, a small community containing 30 source genomes and a large community containing 1000 source genomes. Each community was evaluated under a single coverage design with one coverage profile and a multi coverage design with multiple independently generated coverage profiles. The small multi coverage benchmark used four profiles, and the large multi coverage benchmark used six profiles.

Synthetic paired end reads were generated from the original reference genomes using `RandomReadsMG`[42] at nominal 10x mean depth per independently generated coverage profile. The multi coverage benchmarks were generated independently around the same nominal mean depth so that profile level depth remained comparable while contig level coverage varied across samples. This design produces differential coverage covariance across profiles without allowing methods to separate genomes only by large planned differences in global sequencing depth. Observed mapped depths were verified from the BAM files used in the multi coverage benchmarks. Depth was calculated as mapped records multiplied by 150 bp and divided by the total reference length derived from the BAM header. The large 1000 genome multi coverage benchmark had mapped depths of 10.63x, 10.02x, 10.37x, 10.24x, 9.09x and 9.99x. The small 30-genome multi-coverage benchmark had mapped depths of 10.48x, 10.88x, 13.71x and 9.59x. These values confirm that the profiles were generated near nominal 10x depth while retaining the expected variation from independent simulation, assembly, mapping and alignment filtering.

CAMI II benchmarking

We benchmarked six CAMI II panels with contig-level truth. The retained CAMI II challenge panels included marine, plant and toy mouse gut datasets, together with Human Microbiome Project body site communities from airways, gastrointestinal tract and oral cavity. We selected the HMP collection because the main real metagenome benchmark is weighted toward environmental microbiomes in line with the JGI mission, whereas the HMP panels provide a contrasting host associated benchmark with explicit contig level truth. For each sample, the anonymous gold standard assembly and the official CAMI gsa_mapping.tsv.gz file were used as the reference assembly and contig-to-genome truth set. Scoring against ground truth was performed with BBTools `GradeBins`[38] in CAMI mode. Figure 2 reports paired `CheckM2` estimates and `GradeBins` ground truth scoring for the six displayed CAMI II panels.

Extrachromosomal element benchmark

To evaluate how `QuickBin` handles extrachromosomal elements, we constructed a benchmark from ten public Enterobacteriaceae isolates with corresponding Illumina, Oxford Nanopore Technologies (ONT), and PacBio sequencing data. These isolates come from the De Maio et al.[59] long-read technology comparison dataset, which was generated to compare ONT+Illumina and PacBio+Illumina hybrid assemblies of complex bacterial genomes. Raw reads are available under NCBI BioProject PRJNA422511, and the corresponding hybrid assemblies are available from the Figshare dataset "Hybrid Enterobacteriaceae assemblies using PacBio+Illumina or ONT+Illumina sequencing" (https://doi.org/10.6084/m9.figshare.7649051.v4). The ten isolates used here were RBHSTW-00059, RBHSTW-00122, RBHSTW-00127, RBHSTW-00128, RBHSTW-00131, RBHSTW-00142, RBHSTW-00167, RBHSTW-00189, RBHSTW-00277, and RBHSTW-00350.

For each isolate, we evaluated three assembly and coverage configurations. First, we used the ONT+Illumina hybrid reference assembly from Figshare and mapped the corresponding Illumina paired-end reads to that reference. Second, we used the PacBio+Illumina hybrid reference assembly from Figshare and mapped the matched PacBio reads to that reference. Third, we generated an Illumina-only draft assembly from the matched paired-end Illumina reads with `SPAdes` (v4.2.0)[60] using *--isolate --only-assembler -1 < R1.fastq.gz > -2 < R2.fastq.gz > -t 32 -m 160*, and then mapped the same Illumina reads back to the resulting `SPAdes` contigs. The intention with this design was to keep isolate identity constant while varying whether the binner receives a near-complete hybrid reference or a fragmented short-read assembly, and whether the coverage signal comes from Illumina short reads or PacBio long reads.

Illumina read mapping was performed with BBMap using *ambig = random*, *mateqtag*, *minid = 0.9*, *maxindel = 10*, *threads = 32*, and *-Xmx56g*. PacBio reads were converted from FASTQ to FASTA with `reformat.sh` and mapped with BBMap using *fastareadlen = 600*, *local = t*, *minid = 0.7*, *maxindel = 200000*, *threads = 32*, and *-Xmx56g*. BAM files were coordinate-sorted and indexed with samtools before binning. Binning was run through the same benchmark setup described in the section "Binner parametrization" with sorted BAM input. For the ONT-hybrid and PacBio-hybrid reference configurations, chromosome and plasmid labels were assigned directly from the reference assembly replicons. The longest reference replicon was treated as the host chromosome, and all other reference replicons were treated as plasmid or extrachromosomal replicons for this operational benchmark. For the Illumina SPAdes configuration, fragmented SPAdes contigs were projected back to the ONT-hybrid reference so that the same chromosome/plasmid truth labels could be used. Projection used BBMap local alignment of SPAdes contigs against the ONT-hybrid reference with *local = t*, *ambiguous = best*, *secondary = f*, *minid = 0.85*, and *maxindel = 5000*. Each SPAdes contig was assigned to the best reference replicon by aligned bases and mapping quality; contigs aligning to the longest reference replicon were classified as chromosome-derived, and contigs aligning to other reference replicons were classified as plasmid-derived.

Host/plasmid assignment was then scored from the bin FASTA outputs. For each isolate, platform, and binner, we recorded whether chromosome-derived sequence and plasmid-derived sequence were assigned to the same bin, assigned across multiple bins, or not assigned together. The displayed classes in Supplementary Fig. 4 are: (i) "host and plasmid assigned to one bin", meaning chromosome and plasmid sequence were recovered together in a single bin; (ii) "host and plasmid fragments assigned across multiple bins", meaning at least one plasmid-derived sequence was linked to chromosome-derived sequence but the complete host/plasmid set was split or only partially linked; and (iii) "host and plasmid not assigned together", meaning plasmid-derived

sequence was assigned separately, was unresolved, was absent from the recovered bins, or the binner did not produce usable standardized bins for that platform. This assessment is independent of marker-gene content. Plasmid and other extrachromosomal sequence contributes to the benchmark whenever it is present as a labelled reference replicon or as a SPAdes contig projected to a labelled reference replicon, even if it lacks single-copy marker genes.

Computing environment and benchmarking resource caps

All benchmarks were executed on the NERSC Perlmutter supercomputer (https://www.nersc.gov/). Perlmutter CPU nodes are dual-socket systems with 2x AMD EPYC 7763 (Milan) CPUs, 64 cores per socket (128 physical cores per node), and 512 GB DDR4 memory. Perlmutter GPU nodes are single-socket systems with 1x AMD EPYC 7763 CPU (64 physical cores) and 256 GB DDR4 host memory, paired with 4x NVIDIA A100 GPUs; most GPU nodes provide 40 GB A100 GPUs, and a subset of GPU nodes provide 80 GB A100 GPUs.

Although Perlmutter nodes provide substantially more CPU cores and memory than required for many metagenome binning jobs, we enforced explicit per-run resource caps to ensure fair, deployability-relevant comparisons. For each benchmark run, we limited execution to the CPU core count and RAM specified for that tool and configuration (as reported in the corresponding Results sections and figure panels) by requesting those resources via Slurm and by setting tool thread counts accordingly. All jobs were submitted with a maximum wall-clock time of 24 h (2000× the median `QuickBin` runtime), and runs not completed within this limit were recorded as incomplete for the completion-rate analyses.

Binner parameterization

All binners were executed through a single orchestration layer to enforce uniform inputs, logging, and resource allocation across tools. Each run consumed an assembly FASTA and one sorted BAM file (single coverage) or a directory of sorted BAM files aligned to the same contig set (multi coverage). Binner outputs were collected as per bin FASTA files and standardized into a consistent directory layout for downstream QC and comparisons.

Global thresholds. To make outputs comparable across methods, we applied the same reporting thresholds throughout benchmarking. We used a minimum contig length of 1500 bp and a minimum bin size of 100,000 bp. Among the evaluated binners, `COMEBin` is the only method that does not provide a user-configurable minimum contig length for binning. Consequently, `COMEBin` was run with its default settings, whereas all other binners were run using a 1.5 kbp minimum contig length.

CPU, memory, and runtime control. Each binner was run under explicit CPU and RAM allocations corresponding to the benchmark configuration described in the figure panels, and thread counts were set deterministically from those allocations. All runs were subject to a maximum wall clock time of 24 h (2000× the median `QuickBin` runtime) at the scheduler level. For `QuickBin` (written in Java), Java heap size was capped by a run specific memory budget.

Binner-specific settings

QuickBin. `QuickBin` was run in the default mode and the ultra loose mode ("`QuickBin` [UL]"), as reported in figures. All `QuickBin` runs used SIMD acceleration. To standardize coverage inputs, `QuickBin` runs required sorted BAMs and regenerated coverage profiles every time from the provided alignments.

MetaBAT2. To guarantee fairness and comparability with all other binners, `MetaBAT2` was executed using the `RunMetaBAT` command, which incorporates both the coverage computation and the `MetaBAT2` binning.

VAMB. `VAMB` was run using a fixed random seed for reproducibility. GPU mode was enabled.

SemiBin2 and SemiBin2 [SS]. `SemiBin2` was run in two configurations, standard and self supervised ("SS"). The execution mode was selected based on whether coverage was single sample or multi sample, and GPU mode was enabled.

COMEBin. `COMEBin` was run with GPU mode enabled.

Computational resources measurement

Computational resource consumption was quantified for each bioinformatics tool executed within our benchmarking pipeline using `GNU time` v1.9 (https://www.gnu.org/software/time/) invoked with the verbose flag (-v). This utility was automatically prepended to all tool invocations via wrapper functions to ensure consistent resource tracking across all binning algorithms and quality control steps. For each tool execution, `GNU time` captured the following metrics: user CPU time (seconds spent executing in user mode), system CPU time (seconds spent in kernel mode), elapsed wall-clock time (total real time from process start to completion), percent CPU utilization (ratio of CPU time to wall-clock time), and maximum resident set size (peak physical memory usage in kilobytes).

Statistics and reproducibility

No statistical hypothesis testing was used to compare binners. Benchmark results are reported as descriptive counts, percentages, completion status, elapsed time, and peak memory usage. For real metagenomes, the benchmark included 297 public JGI metagenomes, comprising 275 Illumina assemblies and 22 PacBio assemblies. For each binner and dataset, outputs were evaluated using the same downstream quality assessment and filtering procedures. CAMI II and controlled synthetic benchmarks were evaluated with both `CheckM2` estimates and contig origin ground truth where truth labels were available. Real metagenomes lack contig origin truth, so prokaryotic MAGs were evaluated with `CheckM2` and eukMAGs with `EukCC`. Software versions, resource caps, and database versions are reported in Methods and Supplementary Table S1. Numerical source data for all graphs are provided in Supplementary Data.

Use of AI tools disclosure

The authors acknowledge the use of AI assistants for both code development and manuscript preparation. Algorithmic and documentation contributions (as detailed in the Methods section) were provided by Anthropic's Claude-Sonnet. Structural feedback on the manuscript was additionally provided by Google's Gemini 3 Pro. All AI-assisted work was directed, reviewed, and validated by the human authors.

Reporting summary

Further information on research design is available in the Nature Portfolio Reporting Summary linked to this article.

Data availability

We did not generate new sequencing data for this study. All environmental DNA sequencing samples analyzed were obtained from the Integrated Microbial Genomes and Metagenomes (IMG/M) system. All files are publicly available and immediately accessible for download. Details on the real metagenome samples used in this project are provided in Supplementary Table S2. Metagenome files corresponding to the real metagenome benchmark can be downloaded from the JGI Data Portal at https://data.jgi.doe.gov/ using the direct link to each sample's data reported in the column "data_download_link" in Supplementary Table S2. The U.S. Department of Energy (DOE) Joint Genome Institute (JGI) is a DOE Office of Science User Facility located at Lawrence Berkeley National Laboratory (Berkeley Lab). To accommodate the massive scale of these datasets, the JGI Data Portal offers highly flexible and scalable retrieval methods. Users can download data manually via a web browser, transfer large-scale datasets efficiently using Globus endpoints or Command Line tools, or access the data programmatically via the JGI API. Metagenome files corresponding to the ground truth benchmarking datasets can be downloaded from Zenodo at

https://zenodo.org/communities/quickbin-data. Reference databases included the GTDB-Tk data package for GTDB release r226, the EukCC database v1.2, and the CheckM2 database. Source data for the main and Supplementary Figs. are provided as Supplementary Data.

Code availability

QuickBin is freely distributed as part of the BBTools suite. Source code and installation instructions are available through the BBTools website (https://bbmap.org). This includes links to the GitHub repository, SourceForge image, or prebuilt BBTools Docker image available on Docker Hub, enabling reproducible execution in local, cluster, and cloud environments. BBTools is free for unlimited use under an LBNL open source license. The source code[61] used in this study has been permanently deposited on Zenodo and is publicly available at https://doi.org/10.5281/zenodo.21631948. Details on version, installation method, and running methods for software and databases used in this project can be found in Supplementary Table S1. In brief, software used in this study and corresponding versions were as follows: Binning was performed with QuickBin (BBTools 39.60; deployed via docker image bryce911/bbtools:39.60), MetaBAT2 v2.18, SemiBin2 v2.2.0, COMEBin v1.0.4, and VAMB v5.0.4; all binners and their respective dependencies, including CUDA libraries required for GPU usage, were installed via pixi unless otherwise noted. Quality assessment and filtering used CheckM2 v1.1.0 (installed via pixi) and EukCC v2.1.3 (executed from the container quay.io/microbiome-informatics/eukcc:2.1.3), and gradebins from BBTools (v39.54; deployed via the Docker image at bryce911/bbtools:39.54). Taxonomic assignment used GTDB-Tk v2.4.1 (installed via mamba environment). Read mapping processing used samtools v1.22.1 (installed via pixi).

Received: 4 March 2026; Accepted: 5 August 2026;
Published online: 25 August 2026

References

1. Bowers, R. M. et al. Minimum information about a single amplified genome (MISAG) and a metagenome-assembled genome (MIMAG) of bacteria and archaea. *Nat. Biotechnol.* **35**, 725–731 (2017).
2. Hug, L. A. et al. A new view of the tree of life. *Nat. Microbiol.* **1**, 16048 (2016).
3. Rinke, C. et al. Insights into the phylogeny and coding potential of microbial dark matter. *Nature* **499**, 431–437 (2013).
4. Nayfach, S. et al. A genomic catalog of Earth's microbiomes. *Nat. Biotechnol.* **39**, 499–509 (2021).
5. Sun, Y. et al. gcMeta 2025: a global repository of metagenome-assembled genomes enabling cross-ecosystem microbial discovery and function research. *Nucleic Acids Res.* https://doi.org/10.1093/nar/gkaf1115 (2025).
6. Almeida, A. et al. A unified catalog of 204,938 reference genomes from the human gut microbiome. *Nat. Biotechnol.* **39**, 105–114 (2021).
7. Pasolli, E. et al. Extensive unexplored human microbiome diversity revealed by over 150,000 genomes from metagenomes spanning age, geography, and lifestyle. *Cell* **176**, 649–662.e20 (2019).
8. Bahram, M. et al. Structure and function of the global topsoil microbiome. *Nature* **560**, 233–237 (2018).
9. Sedlar, K., Kupkova, K. & Provaznik, I. Bioinformatics strategies for taxonomy independent binning and visualization of sequences in shotgun metagenomics. *Comput. Struct. Biotechnol. J.* **15**, 48–55 (2017).
10. Dick, G. J. et al. Community-wide analysis of microbial genome sequence signatures. *Genome Biol.* **10**, R85 (2009).
11. Kang, D. D. et al. MetaBAT 2: an adaptive binning algorithm for robust and efficient genome reconstruction from metagenome assemblies. *PeerJ* **7**, e7359 (2019).
12. Wu, Y.-W., Simmons, B. A. & Singer, S. W. MaxBin 2.0: an automated binning algorithm to recover genomes from multiple metagenomic datasets. *Bioinformatics* **32**, 605–607 (2016).
13. Nissen, J. N. et al. Improved metagenome binning and assembly using deep variational autoencoders. *Nat. Biotechnol.* **39**, 555–560 (2021).
14. Pan, S., Zhu, C., Zhao, X.-M. & Coelho, L. P. A deep Siamese neural network improves metagenome-assembled genomes in microbiome datasets across different environments. *Nat. Commun.* **13**, 2326 (2022).
15. Wang, Z. et al. Effective binning of metagenomic contigs using contrastive multi-view representation learning. *Nat. Commun.* **15**, 585 (2024).
16. Almeida, A. et al. A new genomic blueprint of the human gut microbiota. *Nature* **568**, 499–504 (2019).
17. Gurbich, T. A. et al. MGnify genomes: a resource for biome-specific microbial genome catalogues. *J. Mol. Biol.* **435**, 168016 (2023).
18. Schmidt, T. S. B. et al. SPIRE: a Searchable, Planetary-scale mIcrobiome REsource. *Nucleic Acids Res.* https://doi.org/10.1093/nar/gkad943 (2023).
19. Meyer, F. et al. Critical assessment of metagenome interpretation: the second round of challenges. *Nat. Methods* https://doi.org/10.1038/s41592-022-01431-4 (2022).
20. Wang, Z., Huang, P., You, R., Sun, F. & Zhu, S. MetaBinner: a high-performance and stand-alone ensemble binning method to recover individual genomes from complex microbial communities. *Genome Biol.* **24**, 1 (2023).
21. Shaiber, A. & Eren, A. M. Composite metagenome-assembled genomes reduce the quality of public genome repositories. *mBio* **10**, e00725-19 (2019).
22. Han, H., Wang, Z. & Zhu, S. Benchmarking metagenomic binning tools on real datasets across sequencing platforms and binning modes. *Nat. Commun.* **16**, 2865 (2025).
23. Dalla-Torre, H. et al. Nucleotide Transformer: building and evaluating robust foundation models for human genomics. *Nat. Methods* **22**, 287–297 (2025).
24. Nguyen, E. et al. HyenaDNA: long-range genomic sequence modeling at single nucleotide resolution. Preprint at *ArXiv* https://doi.org/10.48550/arXiv.2306.15794 (2023).
25. Nguyen, E. et al. Sequence modeling and design from molecular to genome scale with Evo. *Science* **386**, eado9336 (2024).
26. Rives, A. et al. Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *Proc. Natl. Acad. Sci. USA* **118**, e2016239118 (2021).
27. Lin, Z. et al. Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science* **379**, 1123–1130 (2023).
28. Whang, S. E., Roh, Y., Song, H. & Lee, J.-G. Data collection and quality challenges in deep learning: a data-centric AI perspective. *VLDB J.* **32**, 791–813 (2023).
29. Bileschi, M. L. et al. Using deep learning to annotate the protein universe. *Nat. Biotechnol.* **40**, 932–937 (2022).
30. Jain, M. et al. Biological sequence design with GFlowNets. Preprint at *arXiv* https://doi.org/10.48550/arXiv.2203.04115 (2022).
31. Ji, Y., Zhou, Z., Liu, H. & Davuluri, R. V. DNABERT: pre-trained Bidirectional Encoder Representations from Transformers model for DNA-language in genome. *Bioinformatics* **37**, 2112–2120 (2021).
32. Yan, B. et al. Recent advances in deep learning and language models for studying the microbiome. *Front. Genet.* **15**, 1494474 (2024).
33. Lamurias, A., Sereika, M., Albertsen, M., Hose, K. & Nielsen, T. D. Metagenomic binning with assembly graph embeddings. *Bioinformatics* **38**, 4481–4487 (2022).
34. Uritskiy, G. V., DiRuggiero, J. & Taylor, J. MetaWRAP-a flexible pipeline for genome-resolved metagenomic data analysis. *Microbiome* **6**, 158 (2018).

35. Sieber, C. M. K. et al. Recovery of genomes from metagenomes via a dereplication, aggregation and scoring strategy. *Nat. Microbiol.* **3**, 836–843 (2018).
36. Pan, S., Zhao, X.-M. & Coelho, L. P. SemiBin2: self-supervised contrastive learning leads to better MAGs for short- and long-read sequencing. *Bioinformatics* **39**, i21–i29 (2023).
37. Chklovski, A., Parks, D. H., Woodcroft, B. J. & Tyson, G. W. CheckM2: a rapid, scalable and accurate tool for assessing microbial genome quality using machine learning. *Nat. Methods* **20**, 1203–1212 (2023).
38. Bushnell, B., Bowers, R. M. & Villada, J. C. GradeBins: a comprehensive framework to augment metagenomic bin quality control. Preprint at *bioRxiv* https://doi.org/10.64898/2026.03.11.709163 (2026).
39. Nurk, S., Meleshko, D., Korobeynikov, A. & Pevzner, P. A. metaSPAdes: a new versatile metagenomic assembler. *Genome Res.* **27**, 824–834 (2017).
40. Hofmeyr, S. et al. Terabase-scale metagenome coassembly with MetaHipMer. *Sci. Rep.* **10**, 10689 (2020).
41. Goldfarb, T. et al. NCBI RefSeq: reference sequence standards through 25 years of curation and annotation. *Nucleic Acids Res.* **53**, D243–D257 (2025).
42. Bushnell, B., Schulz, F. & Villada, J. C. Rapid terabase-scale simulation of realistic metagenomes for experimental design and pathogen detection with RandomReadsMG. *bioRxiv* https://doi.org/10.1101/2025.07.18.665570 (2025).
43. Kim, J., Kim, N., Cha, J. H., Ma, J. & Lee, I. Comprehensive benchmarking of metagenomic binning tools reveals key factors for improved genome recovery. *Nat. Commun.* **17**, 3467 (2026).
44. Riley, R. et al. Terabase-scale coassembly of a tropical soil microbiome. *Microbiol. Spectr.* **11**, e0020023 (2023).
45. Parks, D. H. et al. GTDB: an ongoing census of bacterial and archaeal diversity through a phylogenetically consistent, rank normalized and complete genome-based taxonomy. *Nucleic Acids Res.* **50**, D785–D794 (2022).
46. Massana, R. & López-Escardó, D. Metagenome assembled genomes are for eukaryotes too. *Cell Genom.* **2**, 100130 (2022).
47. Kim, C. Y., Ma, J. & Lee, I. HiFi metagenomic sequencing enables assembly of accurate and complete genomes from human gut microbiota. *Nat. Commun.* **13**, 6367 (2022).
48. Bouras, G. et al. How low can you go? Short-read polishing of Oxford Nanopore bacterial genome assemblies. *Microb. Genom*. **10**, 001254 (2024).
49. Sereika, M. et al. Genome-resolved long-read sequencing expands known microbial diversity across terrestrial habitats. *Nat. Microbiol.* **10**, 2018–2030 (2025).
50. Xue, W. et al. LorBin: efficient binning of long-read metagenomes by multiscale adaptive clustering and evaluation. *Nat. Commun.* **16**, 9353 (2025).
51. Beaulaurier, J. et al. Metagenomic binning and association of plasmids with bacterial host genomes using DNA methylation. *Nat. Biotechnol.* **36**, 61–69 (2018).
52. Mane, A., Faizrahnemoon, M., Vinař, T., Brejová, B. & Chauve, C. PlasBin-flow: a flow-based MILP algorithm for plasmid contigs binning. *Bioinformatics* **39**, i288–i296 (2023).
53. Piera Líndez, P. et al. Accurate plasmid reconstruction from metagenomics data using assembly-alignment graphs and contrastive learning. *Nat. Biotechnol.* https://doi.org/10.1038/s41587-026-03005-7 (2026).
54. Chaumeil, P.-A., Mussig, A. J., Hugenholtz, P. & Parks, D. H. GTDB-Tk v2: memory friendly classification with the Genome Taxonomy Database. *Bioinformatics* https://doi.org/10.1093/bioinformatics/btac672 (2022).
55. Parks, D. H. et al. GTDB release 10: a complete and systematic taxonomy for 715,230 bacterial and 17,245 archaeal genomes. *Nucleic Acids Res.* https://doi.org/10.1093/nar/gkaf1040 (2025).
56. Bushnell, B. *BBMap: A Fast, Accurate, Splice-Aware Aligner*. https://www.osti.gov/servlets/purl/1241166 (2014).
57. Chen, I.-M. A. et al. The IMG/M data management and analysis system v.7: content updates and new features. *Nucleic Acids Res.* **51**, D723–D732 (2023).
58. Kolmogorov, M. et al. metaFlye: scalable long-read metagenome assembly using repeat graphs. *Nat. Methods* **17**, 1103–1110 (2020).
59. De Maio, N. et al. Comparison of long-read sequencing technologies in the hybrid assembly of complex bacterial genomes. *Microb. Genom*. **5**, e000294 (2019).
60. Bankevich, A. et al. SPAdes: a new genome assembly algorithm and its applications to single-cell sequencing. *J. Comput. Biol.* **19**, 455–477 (2012).
61. Bushnell, B. & Villada, J. C. Deployable high-fidelity metagenome binning at scale with QuickBin – BBTools 39.60. (Zenodo). https://doi.org/10.5281/ZENODO.21631948 (2026).

Acknowledgements

The work conducted by the U.S. Department of Energy Joint Genome Institute (https://ror.org/04xm1d337), a DOE Office of Science User Facility, is supported by the Office of Science of the U.S. Department of Energy operated under Contract No. DE-AC02-05CH11231. This research used resources of the National Energy Research Scientific Computing Center (NERSC) (https://ror.org/05v3mvq14), a U.S. Department of Energy Office of Science User Facility located at Lawrence Berkeley National Laboratory.

Author contributions

B.B. and J.C.V. designed the overall project. B.B. developed `QuickBin`. B.B. generated the synthetic metagenome data and trained the neural network models. J.C.V. collected and processed metagenome data for the real-metagenome benchmark. J.C.V. installed and troubleshot software to be run on CPU and GPU nodes at NERSC Perlmutter. J.C.V. wrote the binning pipeline for all binners and performed benchmarking on all data. J.C.V. drafted and wrote the manuscript. B.B. wrote the methods describing the `QuickBin` algorithm and the generation of synthetic data. All authors reviewed and approved the manuscript before submission.

Competing interests

The authors declare no competing interests.

Additional information

Supplementary information The online version contains supplementary material available at https://doi.org/10.1038/s42003-026-10782-z.

Correspondence and requests for materials should be addressed to Brian Bushnell or Juan C. Villada.

Peer review information *Communications Biology* thanks Hiroshi Mori and the other, anonymous, reviewer(s) for their contribution to the peer review of this work. Primary Handling Editors: Silvio Waschina and Tobias Goris. A peer review file is available.

Reprints and permissions information is available at http://www.nature.com/reprints

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Open Access This article is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License, which permits any non-commercial use, sharing, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if you modified the licensed material. You do not have permission under this licence to share adapted material derived from this article or parts of it. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by-nc-nd/4.0/.

© The Author(s) 2026