CheckM

Version: 0.7.0
Last Updated: 11 June 2014

Contents

1. About CheckM

CheckM provides a set of tools for assessing the quality of genome recovered from isolates, single cells, or metagenomic data. It provides robust estimates of genome completeness and contamination using lineage-specific sets of ubiquitous single-copy genes and explicitly accounts for gene collocation. Assessment of genome quality can also be examined using plots depicting key genomic characteristics (e.g., GC, coding density) which highlight sequences outside the expected distributions of a typical genome. CheckM also provides tools for identifying genome bins that are likely candidates for merging based on marker set compatibility, similarity in genomic characteristics, and proximity within a reference genome tree.

2. Contact information

CheckM is in active development and we are interested in discussing all potential applications of this software. We encourage you to send us suggestions for new features. Suggestions, comments, and bug reports can be sent to Donovan Parks (donovan.parks [at] gmail.com). If reporting a bug, please provide as much information as possible and a simplified version of the data set which causes the bug. This will allow us to quickly resolve the issue. Issues may also be reported through our GitHub page at: https://github.com/Ecogenomics/CheckM.

3. Citing CheckM

If you use CheckM in your research, please cite:

> Parks DH, Imelfort M, Skennerton CT, Hugenholtz P, Tyson GW. 2014. Assessing the quality of microbial genomes recovered from isolates, single cells, and metagenomes. Submitted to *Genome Research.*

4. Installing CheckM

- Mike: fill this in!

5. Quick Start

CheckM works on a directory of genome bins in FASTA format. By default, CheckM assumes these files end with the extension 'fna', though this can be changed with the –x flag. CheckM consists of a series of commands in order to support a number of different analyses and workflows. If you are in a rush to get started, the standard workflow for CheckM is as follows:

```
> checkm tree <bin folder> <output folder>
> checkm lineage_set <output folder> <marker file>
> checkm analyze <marker file> <bin folder> <output folder>
> checkm qa <marker file> <output folder>
```

Although not strictly necessary, it is often convenient to keep the <output folder> the same throughout. Each of these commands has a number of options you will eventual want to explore.

6. CheckM Command Line Overview

CheckM is executed from the command line and consists of a series of commands in order to support a number of different analyses and workflows. These commands are organized into several related groups:

Lineage-specific marker set:
- tree → Place bins in the reference genome tree
- tree_qa → Assess phylogenetic markers found in each bin
- lineage_set → Infer lineage-specific marker sets for each bin

Taxonomic-specific marker set:
- taxon_list → List available taxonomic-specific marker sets
- taxon_set → Infer taxonomic-specific marker set

Apply marker set to genome bins:
- analyze → Identify marker genes in bins
- qa → Assess bins for contamination and completeness

Common workflows (combines above commands):
- lineage_wf → Runs tree, lineage_set, analyze, qa
- taxonomy_wf → Runs taxon_set, analyze, qa

Bin QA plots:
- bin_qa_plot → Bar plot of bin completeness, contamination, and strain heterogeneity

Reference distribution plots:
- gc_plot → Create GC histogram and delta-GC plot

```
  coding_plot     → Create coding density (CD) histogram and delta-CD plot
  tetra_plot      → Create tetranucleotide distance (TD) histogram and delta-TD plot
  dist_plot       → Create image with GC, CD, and TD distribution plots together

General plots:
  nx_plot         → Create Nx-plots
  len_plot        → Cumulative sequence length plot
  len_hist        → Sequence length histogram
  marker_plot     → Plot position of marker genes on sequences
  par_plot        → Parallel coordinate plot of GC and coverage
  gc_bias_plot    → Plot bin coverage as a function of GC

Sequence subspace plots:
  cov_pca         → PCA plot of coverage profiles
  tetra_pca       → PCA plot of tetranucleotide signatures

Bin exploration and modification:
  merge           → Identify bins with complementary sets of marker genes
  outliers        → Identify outlier in bins relative to reference distributions
  modify          → Modify sequences in a bin
  unique          → Ensure no sequences are assigned to multiple bins

Utility functions:
  unbinned        → Identify unbinned sequences
  coverage        → Calculate coverage of sequences
  tetra           → Calculate tetranucleotide signature of sequences
  profile         → Calculate percentage of reads mapped to each bin
  join_tables     → Join tab-separated value tables containing bin information
  ssu_finder      → Identify SSU (16S/18S) rRNAs in sequences
  bin_compare     → Compare two sets of bins (e.g., from alternative binning methods)
```

For more information on any of these commands type:

```
> checkm COMMAND -h
```

7. Lineage-specific Workflow

The recommended workflow for assessing the completeness and contamination of genome bins is to use lineage-specific marker sets. This workflow consists of 4 mandatory (M) steps and 1 recommended (R) step:

```
(M) > checkm tree <bin folder> <output folder>
(R) > checkm tree_qa <output folder>
(M) > checkm lineage_set <output folder> <marker file>
(M) > checkm analyze <marker file> <bin folder> <output folder>
(M) > checkm qa <marker file> <output folder>
```

The 'tree' command places putative genomes into a reference genome tree. All genomes to be analyzed must reside in a single 'bins' directory. CheckM assumes genome bins are in FASTA format with the extension 'fna', though this can be changed with the –x flag. The 'tree' command can optionally be followed by the 'tree_qa' command which will indicate the number of phylogenetically informative marker genes found in each genome bin along with a taxonomic string indicating its approximate placement in the tree. If desired, genome bins with few phylogenetically marker genes may be removed in order to reduce the computational requirements of the following commands. Alternatively, if only genomes from a particular taxonomic group are of interest these can be moved to a separate directory and analysed separately. The 'lineage_set' command creates a marker file

indicating lineage-specific marker sets suitable for evaluating each genome. This marker file is passed to the 'analyze' command in order to identify marker genes and estimate the completeness and contamination of each genome bin. Finally, the 'qa' command can be used to produce different tables summarizing the quality of each genome bin.

For convenience, the above workflow can be executed in a single step:

```
> checkm lineage_wf <bin folder> <output folder>
```

8. Taxonomic-specific Workflow

In some cases it is convenient to analyze all genome bins with the same marker set. A common example would be a set of genomes from the same taxonomic group. The workflow for using a taxonomic-specific marker set consists of 3 mandatory (M) steps and 1 recommended (R) step:

```
(R) > checkm taxon_list
(M) > checkm taxon_set <rank> <taxon> <marker file>
(M) > checkm analyze <marker file> <bin folder> <output folder>
(M) > checkm qa <marker file> <output folder>
```

The 'taxon_list' command produces a table indicating all taxa for which a marker set can be produced. All support taxa at a given taxonomic rank can be produced by passing 'taxon_list' the '--rank' flag:

```
> checkm taxon_list --rank phylum
```

The 'taxon_set' command is used to produce a marker set for a specific taxon:

```
> checkm taxon_list phylum Cyanobacteria cyanobacteria.ms
```

This marker file produced by the 'taxon_list' command is passed to the 'analyze' command in order to identify marker genes within each genome bin and estimate its completeness and contamination. All putative genomes to be analyzed must reside in a single 'bins' directory. CheckM assumes genome bins are in FASTA format with the extension 'fna', though this can be changed with the –x flag. Finally, the 'qa' command can be used to produce different tables summarizing the quality of each genome bin.

For convenience, the above workflow can be executed in a single step:

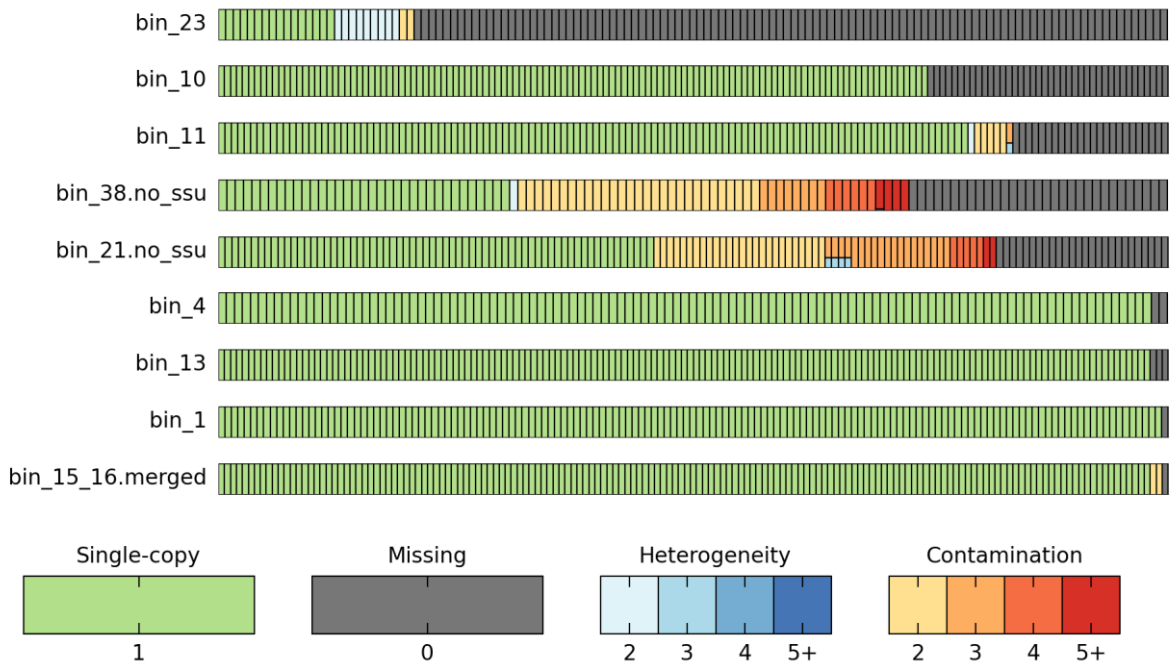
```
> checkm taxonomy_wf <rank> <taxon> <bin folder> <output folder>
```

9. CheckM Plots

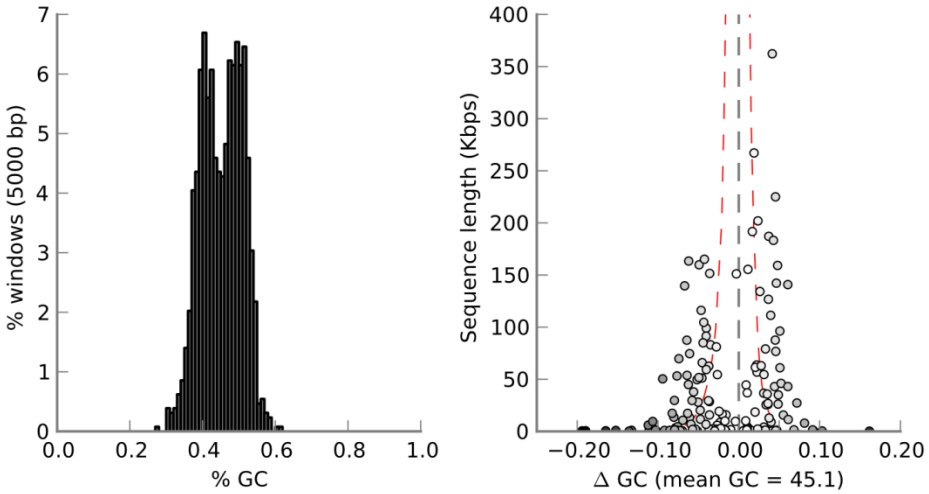
CheckM can produce a number of plots for assessing the quality of genome bins. Here we describe each of these plots and provide an example.

bin_qa_plot: Provides a visual representation of the completeness, contamination, and strain heterogeneity within each genome bin. Bars in green represent markers identified exactly once, while bars in grey represent missing markers. Markers identified multiple times in a genome bin are represented by shades of blue or red depending on the amino acid identity (AAI) between pairs of

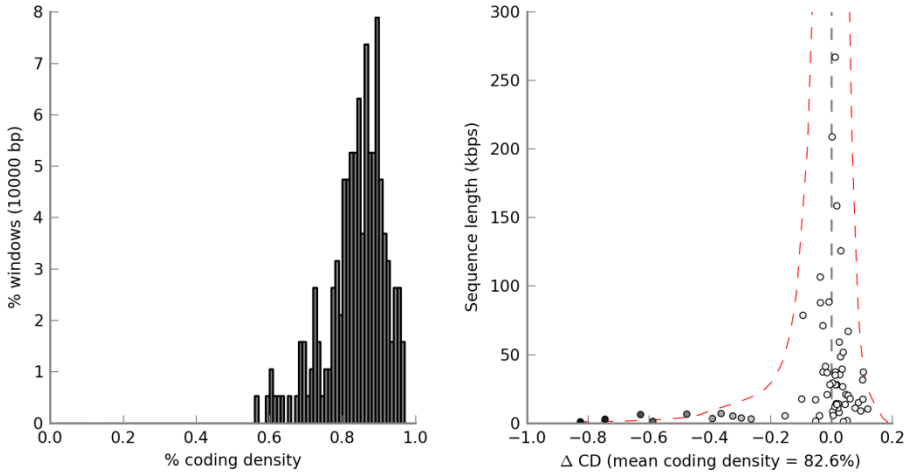
multi-copy genes and the total number of copies present (2-5+). Pairs of multi-copy genes with an AAI ≥90% are indicated with shades of blue, while genes with less amino acid similarity are shown in red. A gene present 3 or more times may have pairs with an AAI ≥90% and pairs with an AAI < 90%.



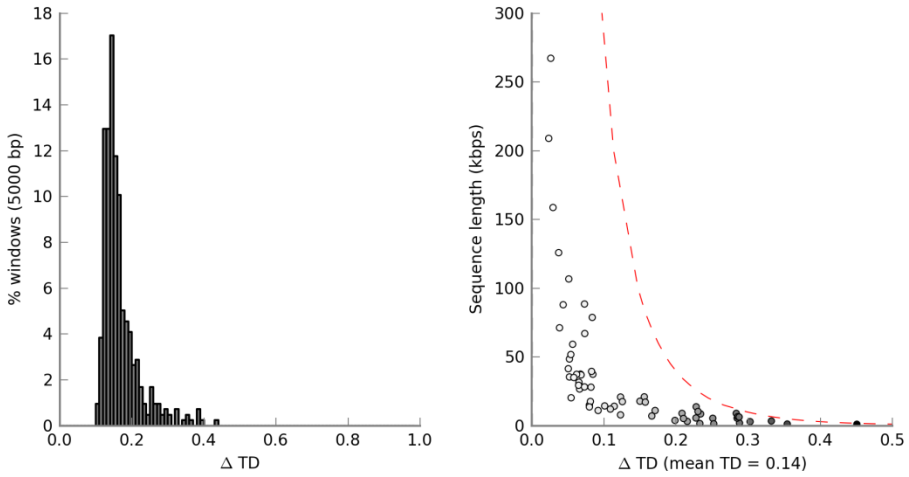
gc_plot: Provides a 3 pane plot suitable for assessing the GC distribution of sequences within a genome bin. The first pane is a histogram of the number of non-overlapping 5 kbp windows with a give percent GC. A typical genome will produce a unimodal distribution. The bimodal distribution in this example suggests this genome bin may be substantially contaminated. The second pane plots each sequence in the genome bin as a function of its change for the average GC of the entire genome (x-axis) and sequence length (y-axis). The dashed red lines indicate the expected deviation from the mean GC as a function of length. This expected deviation is pre-calculated for a set of reference genomes and the exact percentile plotted is provided as an argument to this command.



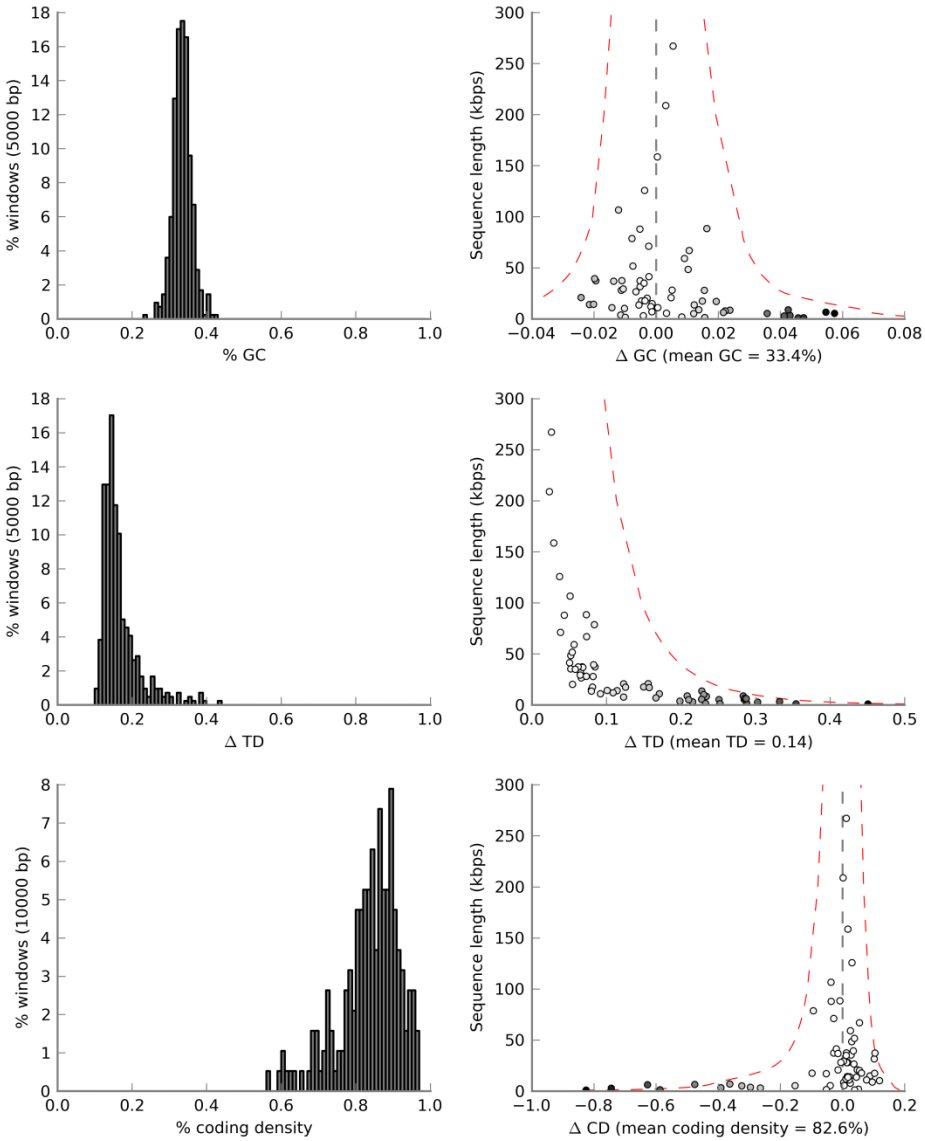
coding_plot: Provides a plot analogous to the *gc_plot* suitable for assessing the coding density of sequences within a genome bin.



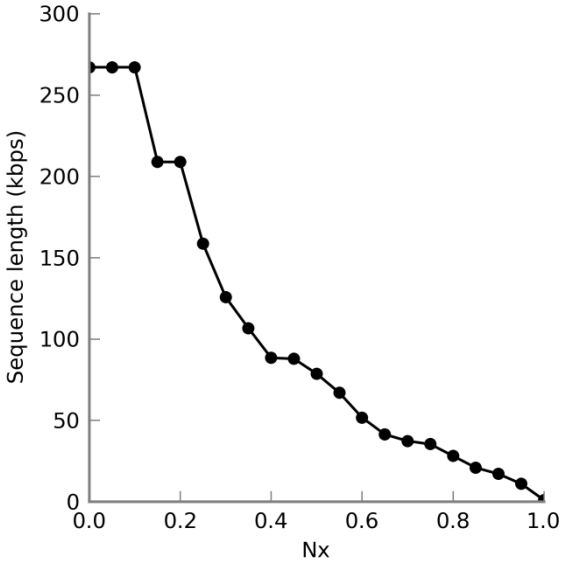
tetra_plot: Provides a plot analogous to the *gc_plot* suitable for assessing the tetranucleotide signatures of sequences within a genome bin. The Manhattan distance is used for determine the different between each sequences tetranucleotide signature and the tetranucleotide signature of the entire genome bin. This plot requires a file indicating the tetranucleotide signature of all sequences within the genome bins. This file can be creates with the 'tetra' command.



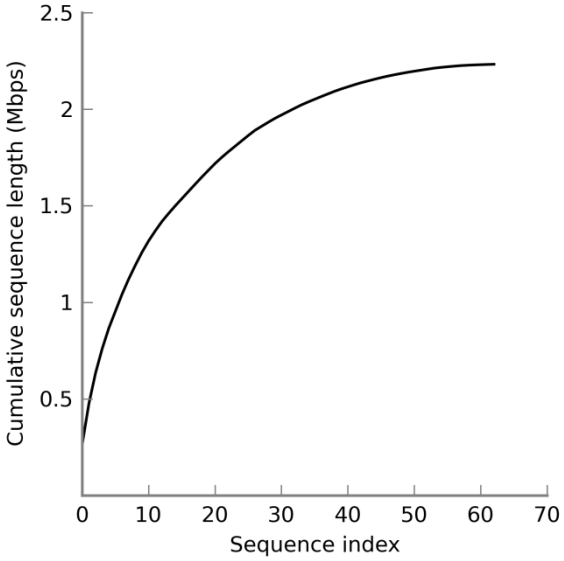
dist_plot: Produces a single figure combining the plots produced by *gc_plot*, *coding_plot*, and *tetra*_plot. This plot requires a file indicating the tetranucleotide signature of all sequences within the genome bins. This file can be creates with the 'tetra' command.



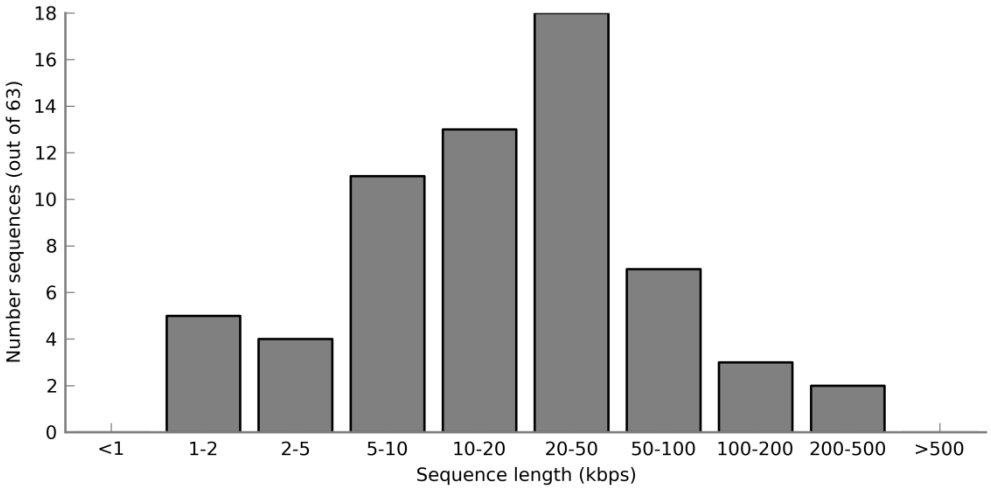
nx_plot: Produces a plot indicating the N*x* value of a genome bin for all values of *x*. This provides a more comprehensive view of the quality of an assembly than simply considering N50.



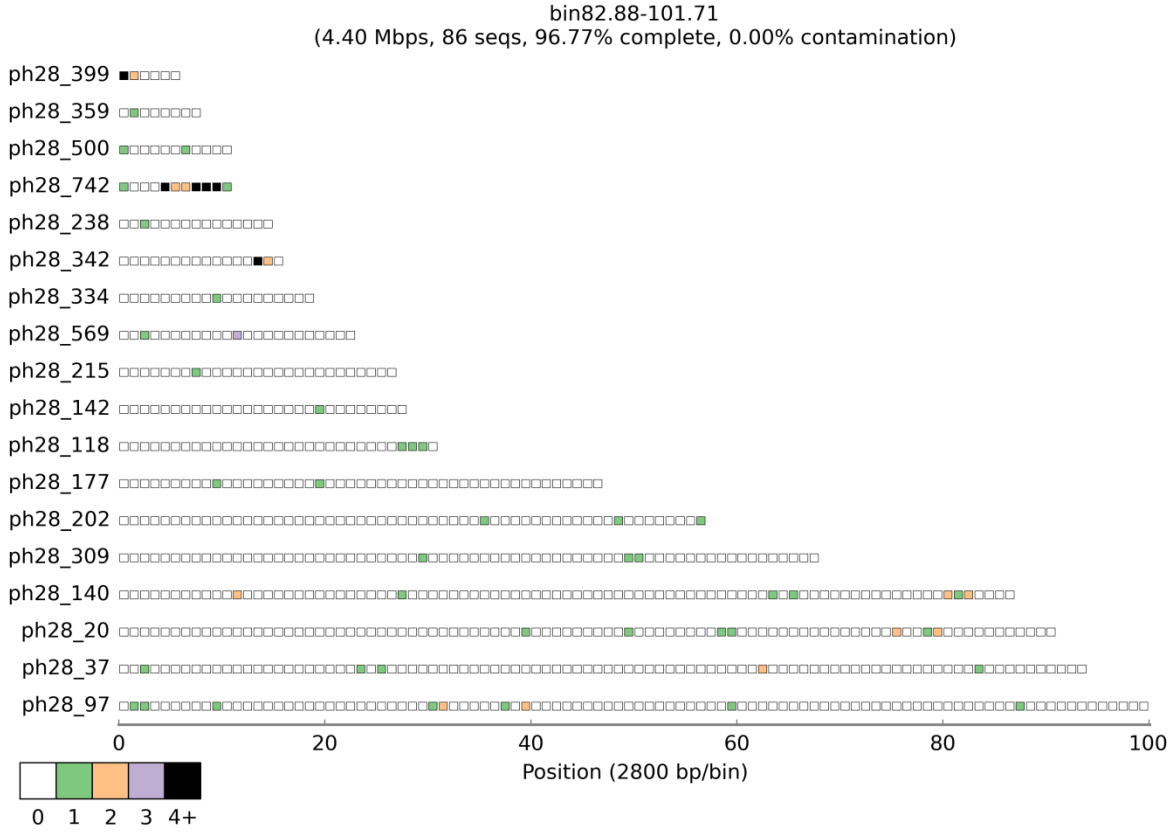
len_plot: Produce a plot of the cumulative sequence length of a genome bin with sequences organized from longest to smallest. This provides additional information regarding the quality of an assembled genome.



len_hist: Produce a histogram of the number of sequences within a genome bin at different sequence length intervals. This provides additional information regarding the quality of an assembled genome.

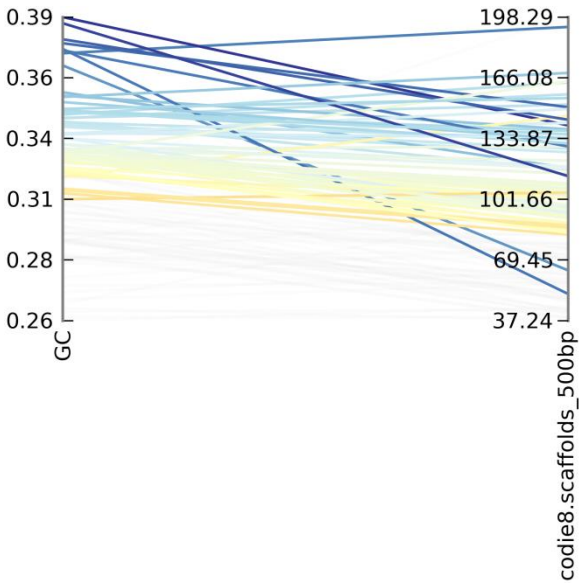


marker_plot: Plots the position of marker genes on sequences within a genome bin. This provides information regarding the extent to which marker genes are collocated. The number of marker genes within a fixed size window (2.8 kbps in this example) is indicated by with different colours. Sequences without any marker genes are not shown.



par_plot: Produces a parallel coordinate plot illustrating the GC and coverage of each sequence within a genome bin. In a typical genome, all sequences will produce a similar path across the plot. Sequences with a divergent path may be contamination. In this example, the scaffolds were obtained from a single metagenomic dataset resulting in a single coverage dimension making it difficult to determine if any sequences might represent contamination. This plot requires a file

indicating the coverage profile of all sequences within the genome bins. This file can be creates with the ‘coverage’ command.



cov_pca: Produces a principal component plot (PCA) of the coverage profile distance between sequences within a putative genome. This plot requires a file indicating the coverage profile of all sequences within the genome bins. This file can be creates with the ‘coverage’ command.

tetra_pca: Produces a principal component plot (PCA) indicating the tetranucleotide distance between sequences within a putative genome. This plot requires a file indicating the tetranucleotide signature of all sequences within the genome bins. This file can be creates with the ‘tetra’ command.

10. Bin Exploration and Modification

unique: Checks each putative genome and ensures no sequences has been assigned to multiple genomes. For most automated binning methods, the assignment of a sequence to multiple putative genomes would indicate a serious binning error. In practice, this command verifies that each sequence name is unique across all putative genome.

merge:

outliers:

modify: This is an experimental command that allows sequences to be added or removed from a putative genome. It is also compatible with the ‘outliers’ function which allows all sequences within a putative genome identified as an outlier to be removed.

11. Additional Utility Functions

CheckM also provides a number of additional commands that may be useful for exploring genome bins. Some of these commands also produce summary statistics of sequences required by specific plotting commands. The utility commands are as follows:

unbinned: Given a set of genome bins along with a FASTA file of sequences determined which of these sequences are not currently contained in a genome bin. This is useful for determine which sequences for an assembly were not binned by an automated binning algorithm

coverage: Produces coverage profiles for all sequences within a set of genome bins. This command requires indexed and sorted BAM files produced with a tool such as BWA. Coverage profiles are required for a number of the plots produced by CheckM.

tetra: Produces tetranucleotide signatures for all sequences within a FASTA file. Tetranucleotide signatures are required for a number of the plots produced by CheckM.

profile: Produces a table indicating the percentage of reads which map to each genome bin. This information is also used to determine the percentage of each genome bin relative to all genome bins under consideration. This is a useful indication of the relative percentage of different populations within a community when the majority of populations are represented by a genome bin. This command requires a file indicating the coverage profile of all sequences within the genome bins. This file can be creates with the 'coverage' command described above.

join_tables: Joins two tab-separated values tables. The first column of each table is used as a unique identifier for joining the tables. A typical use of this command is to join the profile table produced by the 'profile' command with the default output of the 'qa' command.

ssu_finder: Identifies SSU (16S and 18S) rRNA genes residing on sequences if these sequences are contained within a genome bin.

bin_compare: Produces a table indicating the similarity of genome bins produces by two alternative binning methods. This function assumes the same set of sequences was binned by each method. The output is a matrix indicating how binned sequences from the first method map to the genome bins produced by the second method.