MIDAS2

Release 1.0.0

Chunyu Zhao

Sep 08, 2023

CONTENTS

Metagenomic Intra-Species Diversity Analysis System 2 (MIDAS2) is an integrated pipeline for profiling single nucleotide variants (SNVs) and gene copy number variants (CNVs) in shotgun metagenomic reads. MIDAS2 implements the same analyses as the original MIDAS, but re-engineered to addresses the computational challenges presented by increasingly large reference genome databases.

MIDAS2 was developed by Chunyu Zhao and Boris Dimitrov in the Pollard Lab at Chan Zuckerberg Biohub. MIDAS2 expands on the original MIDAS developed by Stephen Nayfach.

For MIDAS2, we have already built two MIDASDBs from large, public, microbial genome databases: UHGG 1.0 and GTDB r202.

Source code is available on GitHub.

Publication is available in Bioinformatics.

CHAPTER

ONE

CONTENTS

1.1 Quickstart

1.1.1 Setup

Install MIDAS2

The fastest way to install all the dependencies MIDAS2 needed is install from YAML file (`midas2.yaml`) using Conda.

On a Linux machine, download a copy of the test data from our GitHub repository release.

```
$ wget https://github.com/czbiohub/MIDAS2/releases/download/v1.0.0/tests.tar.gz
$ tar -zxf tests.tar.gz
$ cd tests
```

Install the dependencies.

```
$ conda env create -n midas2 -f midas2.yml
$ conda activate midas2
$ cpanm Bio::SearchIO::hmmer --force # Temporary fix for Prokka
```

Install MIDAS2.

```
$ pip install midas2
```

Alternative installation procedures are also *described elsewhere*.

Example Data

Assuming you're in the `tests/` directory you just `cd`-ed to, two single-end gzipped FASTQ files are in the folder `tests/reads`. If not, navigate to the `tests` directory

```
$ cd tests
```

1.1.2 Pre-download SCG Genes

Download the universal single copy genes for MIDAS Reference Database (MIDAS DB) of `uhgg` to a new subfolder called `my_midasdb_uhgg`

```
$ midas2 database --init --midasdb_name uhgg --midasdb_dir my_midasdb_uhgg
```

1.1.3 Identify Abundant Species

We'll start by searching for reads that align to single-copy, taxonomic marker genes in order to identify abundant species in each sample.

```
for sample_name in sample1 sample2
do
  midas2 run_species \
      --sample_name ${sample_name} \
      -1 reads/${sample_name}_R1.fastq.gz \
      --midasdb_name uhgg \
      --midasdb_dir my_midasdb_uhgg \
      --num_cores 4 \
      midas2_output
  done
```

1.1.4 Single-nucleotide Variant Analysis

Identify SNVs in Each Sample

We'll next run the single-sample SNV analysis for each sample.

```
for sample_name in sample1 sample2
do
  midas2 run_snps \
    --sample_name ${sample_name} \
    -1 reads/${sample_name}_R1.fastq.gz \
    --midasdb_name uhgg \
    --midasdb_dir my_midasdb_uhgg \
    --num_cores 4 \
    midas2_output
done
```

The pileup summary for `sample1` is written to `midas2_output/sample1/snps/snps_summary.tsv`. This file summarizes the read mapping and pileup results for each of the abundant species determined in the previous step. By default, species are selected based on the filter: `median_marker_coverage > 2`. More details about abundant species selection can be found *here*.

Compute Population SNVs across multiple samples

In order to compute population SNV from multiple single-sample pileup results, we first need to construct a tab-separated **sample manifest file**: `list_of_samples.tsv`.

This file has a column for the `sample_name` and another for `midas_output`, and it is required for multi-sample analyses.

```
echo -e "sample_name\tmidas_outdir" > list_of_samples.tsv
ls reads | awk -F '_' '{print $1}' | awk -v OFS='\t' '{print $1, "midas2_output"}' >>␣
↪list_of_samples.tsv
```

We can take a look at the `list_of_samples.tsv`:

```
cat list_of_samples.tsv
sample_name   midas_outdir
sample1       midas2_output
sample2       midas2_output
```

Based on this output, we can run `merge_snps` and MIDAS2 will know to look at `midas2_output/sample1/snps/snps_summary.tsv` for the `run_snps` output from sample1.

Now we are ready to compute the population SNVs across these two samples:

```
midas2 merge_snps \
  --samples_list list_of_samples.tsv \
  --midasdb_name uhgg \
  --midasdb_dir my_midasdb_uhgg \
  --genome_coverage 0.7 \
  --num_cores 4 \
  midas2_output/merge
```

Users may be interested in the contents of the file `midas2_output/merge/snps_summary.tsv` written in this step.

```
cat midas2_output/merge/snps_summary.tsv
sample_name   species_id      genome_length   covered_bases   total_depth     aligned_
↪reads   mapped_reads    fraction_covered        mean_coverage
sample1       102454  2762447 2322823 15271923        145639  131992  0.841   6.575
sample2       102454  2762447 2322823 15270765        145639  131982  0.841   6.574
```

Other output files and the full output directory structure can be found at *Output Files and Directory Layout*.

1.1.5 Gene Copy-Number Variant Analysis

Identify CNVs in Each Sample

Since building bowtie2 indexes for the species pangenomes takes a long time, we first build the bowtie2 indexes for one species (102454) to a new subfolder `bt2_indexes/`:

```
midas2 build_bowtie2db \
  --midasdb_name uhgg --midasdb_dir my_midasdb_uhgg \
  --species_list 102454 \
  --bt2_indexes_name pangenomes \
  --bt2_indexes_dir bt2_indexes \
  --num_cores 4
```

More information about building your own bowtie2 indexes for either representative genome (repgenome) or pangenome can found *here*.

Now we can run the single-sample CNV analysis for each sample with the existing bowtie2 indexes. The pileup summary for `sample1` will be generated under the directory `midas2_output/sample1/genes/genes_summary.tsv`.

```
for sample_name in sample1 sample2
do
  midas2 run_genes \
    --sample_name ${sample_name} \
    -1 reads/${sample_name}_R1.fastq.gz \
    --midasdb_name uhgg \
    --midasdb_dir my_midasdb_uhgg \
    --prebuilt_bowtie2_indexes bt2_indexes/pangenomes \
    --prebuilt_bowtie2_species bt2_indexes/pangenomes.species \
    --num_cores 4 \
    midas2_output
done
```

Compile CNVs across multiple samples

Same with the population SNV analysis, multi-sample CNV analysis also requires a tab-separated *sample manifest file*.

We can then merge the per-sample CNV results:

```
midas2 merge_genes \
  --samples_list list_of_samples.tsv \
  --midasdb_name uhgg \
  --midasdb_dir my_midasdb_uhgg \
  --num_cores 4 \
  midas2_output/merge
```

Users may be interested in the contents of the file `midas2_output/merge/genes_summary.tsv` written in this step.

```
cat midas2_output/merge/genes_summary.tsv
sample_name   species_id      pangenome_size  covered_genes   fraction_covered       ␣
↪mean_coverage   aligned_reads   mapped_reads    marker_coverage
sample1       102454  129140  4004    0.031   3.495   162476  28611   3.410
sample2       102454  129140  4199    0.033   3.603   169286  34908   3.410
```

Other output files and the full output directory structure can be found at *Output Files and Directory Layout*.

1.2 Installation

MIDAS2 and all its dependencies can be installed in a few ways.

1.2.1 Conda

Install Conda if you have not already. Users can install MIDAS2 and its dependencies with conda package (`midas2`):

```
conda config --set channel_priority flexible
conda install -c zhaoc1 -c anaconda -c bioconda -c conda-forge -c defaults midas2
```

If this installation takes a long time to solve dependencies conflicts, we suggest following the conda install from YAML file approach in the next section.

1.2.2 From Source

Alternatively, users who want the most up-to-date version of the MIDAS code can install from source (dependencies installed with Conda).

On a Linux machins, download a copy of MIDAS2 from our GitHub repository, and install the dependencies. We do not currently support non-Linux environments.

```
$ git clone https://github.com/czbiohub/MIDAS2.git
$ cd MIDAS2

$ conda env create -n midas2.0 -f midas2.yml
$ conda activate midas2
$ cpanm Bio::SearchIO::hmmer --force # Temporary fix for Prokka

$ pip install .
```

Tip: Using the pip `--editable` flag here (`pip install --editable .`) is useful for those wishing to modify the MIDAS source code directly.

1.2.3 Docker

We also provide a pre-built Docker container.

```
docker pull zhaoc1/MIDAS2:latest
docker run --volume "/home/ubuntu/.aws":"/root/.aws":ro --rm -it MIDAS2:latest
```

We've also included integration tests, which can be run using the provided script

```
$ bash tests/test_analysis.sh 8
```

This will run an example analysis with 8 cores, and will verify that all the dependencies are correctly installed and that all analysis modules of MIDAS2 can run properly. We recommend running this after installing MIDAS2 from source.

1.3 MIDAS2 Overview

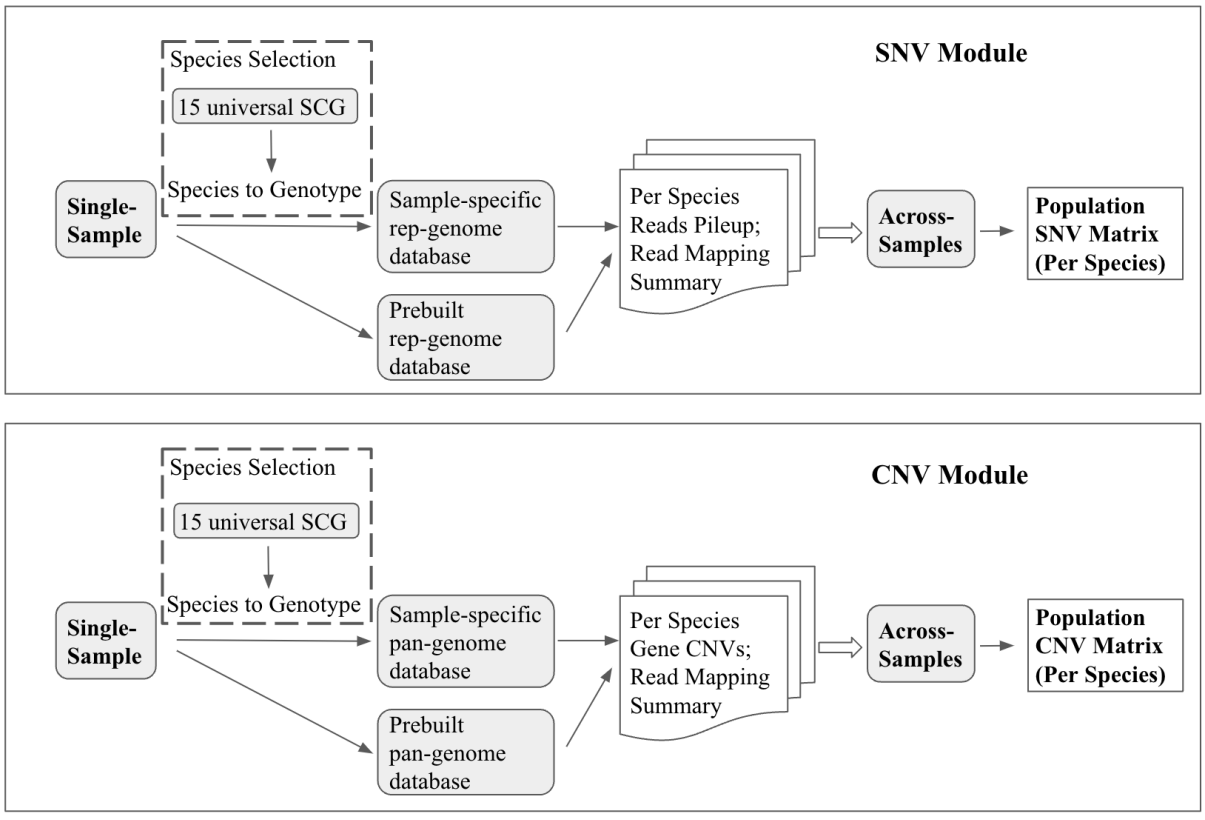
MIDAS2 was developed to address the computational challenges presented by increasingly large reference genome databases. MIDAS2 implements the same analyses as the original MIDAS tool, but re-engineered to

1. host multiple MIDAS Reference Database (MIDAS DB) from public microbial genome collections
2. enable thousands of metagenomic samples to be efficiently genotyped.

MIDAS2 contains two analysis modules focused on:

1. Single nucleotide variants (SNV): *SNV module*
2. Pan-genome copy number variation (CNV): *CNV module*

Each module includes two sequential steps: single-sample analysis and cross-samples merging.



1.3.1 MIDAS Reference Database

MIDAS2, as any reference-based strain-level genomic variation analysis pipeline, presupposes a reference database construction step has already taken place. The MIDAS Reference Database (MIDASDB) refers to a set of custom files with database information needed to run MIDAS2.

The original MIDAS provided a default bacterial reference databases (see Figure 1): MIDAS DB v1.2 was constructed from a collection of 5,952 bacterial species clusters representing 31,007 high-quality bacterial genomes.

However, in the past few years, the number of sequenced microbial genomes has increased a lot, in particular with the addition of metagenome-assembled genomes (MAGs) sequenced from varied habitats. Therefore, it was necessary to

update MIDAS DBs in MIDAS2 accordingly. On the other hand, processing the large amount of available genome sequences poses a significant computational challenge.

For MIDAS2, instead of generating the species clusters from scratch, we take advantage of two published collections of prokaryotic genome databases, and we build a MIDASDB for each one.

More information about these genome collections can be found in *Download MIDASDB*.

1.3.2 Command-line Usage

MIDAS2 operates through a command-line interface (CLI). This interface enables reproducible analyses and allows MIDAS to be integrated into workflow management frameworks.

Common CLI Options

Output Directory

MIDAS2 writes its outputs to a user-specified root directory, which is always passed as a mandatory argument to each of the MIDAS2 analyses command.

For example, in *Quickstart*, `midas2_output` is the chosen output directory, and all analyses steps operate within it.

Single-sample Commands

The three single-sample commands (`run_species`, `run_snps` and `run_genes`) share a number of command-line flags.

Sample Name

Users need to chose a unique `sample_name` per sample, and together with the output directory, `midas2_output/sample_name` constitutes the unique output directory for single-sample analyses.

Input Reads

The FASTA/FASTQ file containing single-end or paired-ends sequencing reads needs to be passed via the arguments as:

```
-1 /path/to/R1.fastq.gz  -2 /path/to/R2.fastq.gz
```

Across-samples Commands

A tab-separated sample manifest file listing the `sample_name` and full path of the single-sample root output directory `midas_output` is required for across-samples analyses.

Users need to pass the path of this file to the command-line argument `--sample_list`. For example, in the Quickstart, we passed as following: `--sample_list list_of_samples.tsv`.

A template is shown here:

sample_name	*midas_outdir*
sample1	/home/ubuntu/MIDAS2.0/tests/midas2_output
sample2	/home/ubuntu/MIDAS2.0/tests/midas2_output

MIDAS Reference Database

For all MIDAS2 analysis, users need to choose

1. a valid precomputed MIDASDB name (uhgg, gtdb) as `--midasdb_name`
2. a valid local path for the downloaded MIDASDB `--midasdb_dir`.

For example, in *QuickStart*, we downloaded the SCG marker database for `--midasdb_name uhgg` into `--midasdb_dir my_midasdb_uhgg`.

Others Parameters

Users can set the `--num_cores` to the number of physical cores to use: e.g. `--num_cores 16`.

And all MIDAS2 analyses can print out the full help message and exit by `-h` or `--help`.

1.4 Module: Species Selection

Reference-based metagenotyping depends crucially on the choice of reference sequences. Microbiome data may contain hundreds of species in one sample, and an ideal reference database is both representative and comprehensive in terms of the abundant species in the sample. A badly chosen reference may suffer both from ambiguous mapping of reads to two or more sequences (which may lead to reads being filtered out post-alignment due to low uniqueness) or spurious cross-mapping to incorrect sequences (which leads to metagenotype errors). Therefore, a typical MIDAS2 workflow starts with a species selection step, which customizes the MIDASDB to only include the genomes of sufficiently abundant species in each sample.

Contents

1.4.1 Single-Sample Analysis

MIDAS2 estimates species coverage by profiling the coverage of 15 universal, single copy, marker genes (SCGs, 15 per species), to quickly determine which species are abundant in the sample.

Warning: This is designed *only* to select species with sufficient coverage in each sample. It is not intended to quantify species abundance.

(In this document, we continue to use the *data* from the Quickstart as an example.)

```
midas2 run_species \
    --sample_name sample1 \
    -1 reads/sample1_R1.fastq.gz \
    --midasdb_name uhgg \
    --midasdb_dir my_midasdb_uhgg \
    --num_cores 4 \
    midas2_output
```

Tip: Single-sample analysis step can be parallelized over samples (e.g. xargs)

Note: The first time `run_species` is used, MIDAS2 will automatically download the marker gene database.

Warning: (Race condition) If starting multiple calls to `run_species` simultaneously, be sure that the marker gene database has already been *downloaded*. Otherwise multiple, redundant downloads may be started.

1.4.2 Cross-Sample Merging

We now run the `merge_species` command to merge the single-sample species profiling results for the samples listed our *samples_list*.

```
midas2 merge_species \
  --samples_list list_of_samples.tsv \
  --min_cov 2 \
  midas2_output/merge
```

The `--min_cov` flag defines the minimum `median_marker_coverage` for estimating species prevalence, which is output as the `sample_counts` statistic. See below.

1.4.3 Key Outputs

Single-Sample

For each sample, the primary output of the `run_species` command is a file containing information about species detected in the reads (e.g.) `midas2_output/sample1/species/species_profile.tsv` This file describes the coverage of each species' marker genes in the sample. Species are sorted in decreasing order of `median_marker_coverage`. Only species with more than two marker genes covered with more than two reads (a very low bar) are reported in this file.

species_id	*marker_read_counts*	*median_marker_coverage*	*marker_coverage*	*marker_relative_abundance*	*unique_fraction_covered*
102337	4110	28.48	28.91	0.30	1.00
102506	734	4.98	4.98	0.05	0.93

Where the columns have the following meaning:

```
species_id:                six-digit species id
marker_read_counts:        total mapped read counts
median_marker_coverage:    median coverage of the 15 SCGs
marker_coverage:           mean coverage of the 15 SCGs
marker_relative_abundance: computed based on ``marker_coverage``
unique_fraction_covered:   the fraction of uniquely mapped SCGs genes
```

Downstream commands (`run_snps` and `run_genes`) use the `median_marker_coverage` and/or `unique_fraction_covered` to select sufficiently abundant species. See below.

Across-Samples

The primary output of the merging step is the file `midas2_output/merge/species/species_prevalence.tsv`.

species_id	*median_abundance*	*mean_abundance*	*median_coverage*	*mean_coverage*	*sample_counts*
102337	0.186	0.186	16.205	16.205	2
102506	0.035	0.035	2.967	2.967	2

Where the columns have the following meaning:

```
species_id:       six-digit species id
median_abundance: median marker_relative_abundance across samples
mean_abundance:   mean marker_relative_abundance across samples
median_coverage:  median median_marker_coverge across samples
mean_coverage:    mean median_marker_coverge across samples
sample_counts:    number of samples with median_marker_coverge >= min_cov
```

MIDAS2 also writes two species-by-sample matrices in the output directory: `midas2_output/merge/species`. Median marker coverage, and unique fraction covered are written to `midas2_output/merge/species/species_marker_median_coverage.tsv` and `midas2_output/merge/species/species_unique_fraction_covered.tsv`, respectively

1.4.4 Species Selection in Downstream Modules

In a standard SNV/CNV workflow, only sufficiently abundant species in the restricted species profile will be included to build representative genome (rep-genome) or pan-genome index and further to be genotyped. By default, both the `run_snps` and `run_genes` commands perform a species selection step. Both commands therefore assume that `run_species` has already been carried out for each sample.

Two flags, `--select_by` and `--select_threshold`, determine which species are selected:

- `--select_by` followed by a comma separated list of column names in `midas2_output/species/species_profile.tsv`
- `--select_threshold` followed by a comma-separated list of threshold values for selection.

For most analyses we recommend using the combination of `median_marker_coverage` > `2X` and `unique_fraction_covered` > `0.5`:

```
--select_by median_marker_coverage,unique_fraction_covered --select_threshold=2,0.5
```

Some users may wish to genotype low abundance species and should adjust the parameters accordingly:

```
--select_by median_marker_coverage,unique_fraction_covered --select_threshold=0,0.5
```

Alternatively, users can directly pick a list of species using the `--species_list` option. It is worth noting that the species in the provided species list are still subject to the `--select_threshold` restriction. Users can set `--select_threshold=-1` to escape species selection filters based on the species profiling:

```
--species_list 102337,102506 --select_threshold=-1
```

All the species passing the species selection filters will be genotyped.

Having finished the species selection step, we can now go to the SNV or CNV modules, depending on the scientific aims.

1.5 Module: Single Nucleotide Variant Analysis

The Single Nucleotide Variant (SNV) module has two commands:

1. Single-sample allele tallying with the `run_snps` command;
2. SNV calling across all the samples, with `merge_snps` command.

The first step can be run in parallel. We assume users have already completed the *species module* and have a species profile (e.g. `midas2_output/sample1/species/species_profile.tsv`) ready for each sample. Alternatively, advanced users can pass a *pre-built representative genome index* ready for single-sample SNV analysis.

Contents

1.5.1 Single-Sample Analysis

Conceptually, a typical invocation of the `run_snps` command proceeds by

1. selecting sample-specific abundant species based on statistics calculated in the *species module*;
2. compiling these representative genomes and building a sample-specific bowtie2 index;
3. mapping reads to this index with bowtie2;
4. outputting a read mapping summary and pileup result for each representative genome (i.e. each species).

MIDAS2 purposely **holds off any filtering** or identification of variant positions until the subsequent cross-sample analysis.

(In this document, we continue to use the *data* from the Quickstart as an example.)

```
midas2 run_snps \
  --sample_name sample1 \
  -1 reads/sample1_R1.fastq.gz \
  --midasdb_name uhgg \
  --midasdb_dir my_midasdb_uhgg \
  --select_by median_marker_coverage,unique_fraction_covered \
  --select_threshold=2,0.5 \
  --num_cores 8 \
  midas2_output
```

See the documentation on *selecting abundant species* for more information about the `--select_by` and `--select_threshold` flags.

Tip: This step can be parallelized over samples (e.g. using shell background processes).

Note: In MIDAS2, `run_snps` can automatically download the reference genomes for the selected species.

Warning: (Race condition) If starting multiple calls to `run_snps` simultaneously, be sure that reference genomes have already been *downloaded*. Otherwise multiple redundant downloads may be started.

1.5.2 Cross-Samples Analysis

After running all samples individually in this way, users can then compute population SNVs across samples using the `merge_snps` command.

```
midas2 merge_snps \
  --samples_list list_of_samples.tsv \
  --midasdb_name uhgg \
  --midasdb_dir my_midasdb_uhgg \
  --num_cores 8 \
  midas2_output/merge
```

1.5.3 Key Outputs

Single-Sample

Unlike the *species* and *CNV* modules, the single-sample outputs from the SNV module are less interesting than the merged results (at least with the default mode).

Users may, however, find several files useful.

A summary of read alignment and pileups for each of the genomes included in the (usually sample-specific) bowtie2 index is reported in `midas2_output/sample1/snps/snps_summary.tsv`.

species_id	*genome_length*	*covered_bases*	*total_depth*	*aligned_reads*	*mapped_reads*	*fraction_covered*	*mean_coverage*
102506	5339468	2373275	8045342	468667	224553	0.444	3.390
102337	2749621	2566404	47723458	1479479	1010530	0.933	18.595

Where each columns has the following meaning:

```
species_id:       six-digit species id
genome_length:    genome length
covered_bases:    number of bases covered by at least one post-filtered reads
total_depth:      total read depth across all covered_bases
aligned_reads:    total read counts across covered_bases before post-alignment filter
mapped_reads:     total read counts across covered_bases after post-alignment filter
fraction_covered: fraction of covered_bases (aka horizontal genome coverage)
mean_coverage:    mean read depth across all covered_bases (aka vertical genome coverage)
```

For each sample and species—e.g. here sample1 and species 102506 (*E. coli*)—the per-species read pileup is found in `midas2_output/sample1/snps/102506.snps.tsv.lz4`. Positions are filtered to only sites in the reference genome covered by at least two reads.

Note: Large output files are compressed with LZ4 to minimize storage requirements.

When uncompressed, the contents of this file should look like the following CSV:

ref_id	*ref_pos*	*ref_allele*	*depth*	*count_a*	*count_c*	*count_g*	*count_t*
gnl|Prokka|UHGG144544_1	881435	T	11	0	0	0	11
gnl|Prokka|UHGG144544_1	881436	T	13	0	5	0	8
gnl|Prokka|UHGG144544_1	881437	T	12	0	6	0	6

Where the columns have the following meaning:

```
ref_id:     scaffold/contig id
ref_pos:    reference position
ref_allele: reference nucleotide
depth:      number of post-filtered reads
count_a:    post-filtered read counts of A allele
count_c:    post-filtered read counts of C allele
count_g:    post-filtered read counts of G allele
count_t:    post-filtered read counts of T allele
```

Across-Samples

A number of outputs result from the multiple samples population SNV analysis.

A merged pileup summary is found in `midas2_output/merge/snps/snps_summary.tsv`.

sample_name	*species_id*	*genome_length*	*covered_bases*	*total_depth*	*aligned_reads*	*mapped_reads*	*fraction_covered*	*mean_coverage*
sample1	100122	2560878	2108551	10782066	248700	207047	0.823	5.113
sample2	100122	2560878	2300193	39263110	1180505	820736	0.898	17.069

The reported columns from `genome_length` to `mean_coverage` are the same as from the single-sample SNV summary.

For each species, information about SNVs identified across samples is written to `midas2_output/merge/snps/102506.snps_info.tsv.lz4`.

site_id	*major_allele*	*minor_allele*	*sample_counts*	*snp_type*	*rc_A*	*rc_C*	*rc_G*	*rc_T*	*sc_A*	*sc_C*	*sc_G*	*sc_T*	*locus_type*	*gene_id*	*site_type*	*amino_acids*
gnl|Prokka|UHGG002587_1|3436026	A	G	2	bi	26	10	0	0	2	2	0	0	CDS	UHGG002587_00058	1D	T,T,T,T
gnl|Prokka|UHGG002587_1|3839904	G	T	1	bi	4	0	11	45	0	0	2	2	IGR	None	None	None

Where columns have the following meaning:

```
site_id:       unique site id, composed of ref_id|ref_pos|ref_allele
major_allele:  most common/prevalent allele in metagenomes
minor_allele:  second most common/prevalent allele in metagenomes
sample_counts: number of relevant samples where metagenomes is found
snp_type:      the number of alleles observed at site (mono,bi,tri,quad)
rc_A:          accumulated read counts of A allele in metagenomes
rc_C:          accumulated read counts of C allele in metagenomes
rc_G:          accumulated read counts of G allele in metagenomes
rc_T:          accumulated read counts of T allele in metagenomes
sc_A:          accumulated sample counts of A allele in metagenomes
sc_C:          accumulated sample counts of C allele in metagenomes
sc_G:          accumulated sample counts of G allele in metagenomes
sc_T:          accumulated sample counts of T allele in metagenomes
locus_type:    CDS (site in coding gene), RNA (site in non-coding gene), IGR (site in␣
↪intergenic region)
gene_id:       gene identified if locus type is CDS, or RNA
site_type:     indicates degeneracy: 1D, 2D, 3D, 4D
amino_acids:   amino acids encoded by 4 possible alleles
```

A site-by-sample minor allele frequency matrix is written to `midas2_output/merge/snps/102506.snps_freq.tsv.lz4`.

site_id	*sample1*	*sample2*
gnl|Prokka|UHGG000587_11|83994|T	0.692	0.837
gnl|Prokka|UHGG000587_14|34360|A	0.300	0.269

A site-by-sample read depth matrix is written to `midas2_output/merge/snps/102506.snps_freq.tsv.lz4`.

Note: This table only accounts for the alleles matching the population major and/or minor allele. Other bases are dropped.

site_id	*sample1*	*sample2*
gnl|Prokka|UHGG000587_11|83994|T	13	43
gnl|Prokka|UHGG000587_14|34360|A	10	26

1.5.4 Advanced SNV Calling

Single-Sample Post-alignment Filter

Users can adjust post-alignment filters via the following command-line options (default values indicated):

- `--mapq >= 10`: discard read alignment with alignment quality < 10
- `--mapid >= 0.94`: discard read alignment with alignment identity < 0.94
- `--aln_readq >= 20`: discard read alignment with mean quality < 20
- `--aln_cov >= 0.75`: discard read alignment with alignment coverage < 0.75
- `--aln_baseq >= 30`: discard bases with quality < 30
- `--paired_only`: only recruit properly aligned read pairs for post-alignment filter and pileup
- `--fragment_length 5000`: maximum fragment length for paired-end alignment. Incorrect fragment length would affect the number of proper-aligned read pairs

Single-Sample Advanced SNV Calling

In recognition of the need for single-sample variant calling, we provided `--advanced` option to users for single-sample variant calling for all the species in the rep-genome index with `run_snps` command.

In the `--advanced` mode, per-species pileup results will also report major allele and minor allele for all the genomic sites covered by at least two post-filtered reads, upon which custom variant calling filter can be applied by the users. Users are advised to use the setting `--ignore_ambiguous` to avoid falsely calling major/minor alleles for sites with tied read counts.

```
midas2 run_snps
  --sample_name sample1 \
  -1 reads/sample1_R1.fastq.gz \
  --midasdb_name uhgg \
  --midasdb_dir my_midasdb_uhgg \
  --select_by median_marker_coverage,unique_fraction_covered \
```

(continues on next page)

(continued from previous page)

```
  --select_threshold=2,0.5 \
  --fragment_length 2000 --paired_only \
  --advanced --ignore_ambiguous \
  --num_cores 8
  midas2_output
```

Expected Output

In the `--advanced` mode, per-species pileup results will include five additional columns of the major/minor allele for all the covered genomic sites.

ref_id	*ref_pos*	*ref_allele*	*depth*	*count_a*	*count_c*	*count_g*	*count_t*	*major_allele*	*minor_allele*	*major_allele_freq*	*minor_allele_freq*	*allele_counts*
gnl|Prokka|UHGG814554_1	814554	T	11	0	0	0	11	T	T	1.000	0.000	1
gnl|Prokka|UHGG814564_1	814564	T	13	0	5	0	8	T	C	0.615	0.385	2
gnl|Prokka|UHGG814574_1	814574	T	12	0	6	0	6	C	T	0.500	0.500	2

- `major_allele`: the allele with the most read counts
- `minor_allele`: the allele with the 2nd most read counts; same with major_allele if only one allele is observed
- `major_allele_freq`: allele frequency of `major_allele`
- `minor_allele_freq`: allele frequency of `minor_allele`; 0.0 if only one allele is observed
- `allele_counts`: number of alleles observed

Adjust Population SNV Filters

Advanced users can refer to *this page* for understanding the compute of population SNV. The species, sample, and site filters for the across-samples SNV calling can be customized with command-line options. For example,

- We can select species with `horizontal coverage > 40%`, `vertical coverage > 3X` and present in more than 2 relevant samples:

```
--genome_coverage 0.4 --genome_depth 3 --sample_counts 2
```

- We can apply the following site selections: only consider site with `read depth >= 5`, and `read depth <= 3 * genome_depth`, and the minimal allele frequency to call an allele present is 0.05.

```
--site_depth 5 --site_ratio 3 --snp_maf 0.05
```

- We can only report populations SNV meeting the following criteria: bi-allelic, common population SNV (present in more than 80% of the population) from the protein coding genes based on accumulated sample counts.

```
--snp_type bi --snv_type common --site_prev 0.8 --locus_type CDS --snp_pooled_method␣
↪prevalence
```

Now we can put all the above-mentioned filters in one *merge_snps* command:

```
midas2 merge_snps
  --samples_list list_of_samples.tsv \
  --midasdb_name uhgg \
```

(continues on next page)

(continued from previous page)

```
  --midasdb_dir my_midasdb_uhgg \
  --genome_coverage 0.4 --genome_depth 3 --sample_counts 2 \
  --site_depth 5 --site_ratio 3 --snp_maf 0.05 \
  --snp_type bi --snv_type common --site_prev 0.8 --locus_type CDS --snp_pooled_method␣
↪prevalence \
  --num_cores 8 \
  midas2_output/merge
```

1.6 Module: Copy Number Variant Analysis

Similar to the SNV module, the Copy Number Variant (CNV) module has two commands:

1. Single-sample, pan-genome copy number quantification with the `run_genes` command;
2. Merging these results across all samples with the `merge_genes` command.

The first step can be run in parallel. We assume users have already completed the *species module* and have a species profile (e.g. `midas2_output/sample1/species/species_profile.tsv`) ready for each sample. Alternatively, advanced users can pass a pre-built pangenome bowtie2 index.

Contents

1.6.1 Single-Sample Analysis

Conceptually, a typical invocation of the `run_genes` command proceeds by

1. selecting sample-specific abundant species based on statistics calculated in the *species module*;
2. compiling species pan-genomes and building a sample-specific bowtie2 index;
3. mapping reads to this index with bowtie2;
4. outputting a read mapping summary and copy number profile for each pan-genome (i.e. each species).

For each species, copy numbers for each gene are estimated as follows:

1. For each gene in the species pangenome, read alignment metrics, e.g `mapped_reads` and `mean_coverage`, are computed from the bowtie2 results;

2. species-level coverage is calculated as the median read coverage of the 15 universal single-copy taxonomic marker genes.
3. gene copy number (and presence/absence) is estimated by normalizing the gene coverage by the species coverage.

(In this document, we continue to use the *data* from the Quickstart as an example.)

```
midas2 run_genes \
  --sample_name sample1 \
  -1 reads/sample1_R1.fastq.gz \
  --midasdb_name uhgg \
  --midasdb_dir my_midasdb_uhgg \
  --select_by median_marker_coverage,unique_fraction_covered \
  --select_threshold=2,0.5 \
  --num_cores 8 \
  midas2_output
```

See the documentation on selecting abundant species for more information about the `--select_by` and `--select_threshold` flags.

Tip: This step can be parallelized over samples (e.g. using shell background processes).

Note: In MIDAS2 `run_genes` can automatically download gene collections for the selected species.

Warning: (Race condition) If starting multiple calls to `run_genes` simultaneously, be sure that the gene collections have already been *downloaded*. Otherwise multiple redundant downloads may be started. TODO: Link to the preload instructions here.

1.6.2 Cross-Samples Merging

Having run the single-sample CNV analysis for all the samples listed in the `list_of_samples.tsv`, users next can merge the results and product a summary using the `merge_genes` command.

```
midas2 merge_genes \
  --samples_list list_of_samples.tsv \
  --midasdb_name uhgg \
  --midasdb_dir my_midasdb_uhgg \
  --num_cores 8 \
  midas2_output/merge
```

1.6.3 Key Outputs

Single-Sample

For each sample (e.g. here sample1) a summary of read alignment and CNV calling across all analyzed species is written to `midas2_output/sample1/genes/genes_summary.tsv`.

species_id	*pangenome_size*	*covered_genes*	*fraction_covered*	*mean_coverage*	*aligned_reads*	*mapped_reads*	*marker_coverage*
102337	15578	4468	0.287	16.213	1650361	450353	20.213
102506	731186	4733	0.006	3.803	681335	37272	2.140

Where each columns has the following meaning:

```
species_id:       six-digit species id
pangenome_size:   number of centroids (non-redundant genes) in the species pangenome
covered_genes:    number of centroids covered with at least one post-filtered read
fraction_covered: fraction of covered_genes over pangenome_size
mean_coverage:    average read depth across covered_genes
aligned_reads:    total number of aligned reads before post-alignment filter
mapped_reads:     total number of aligned reads after post-alignment filter
marker_coverage:  average read depth across 15 universal SCGs in the species pangenome
```

Copy-number estimates are written to `midas2_output/sample1/genes/102506.genes.tsv.lz4` and include all genes covered by at least two reads.

Note: Large output files are compressed with LZ4 to minimize storage requirements.

gene_id	*gene_length*	*aligned_reads*	*mapped_reads*	*mean_coverage*	*fraction_covered*	*copy_number*
UHGG143901_00483	555	14	6	2.961538	0.234234	1.384035
UHGG143901_03589	384	103	57	32.840708	0.294271	15.347667
UHGG143902_04031	207	9	2	1.737500	0.386473	0.811997

Where columns have the following meaning:

```
gene_id:          centroid id in the species pan-genome
gene_length:      gene length
aligned_reads:    number of aligned reads to gene_id before post-alignment filter
mapped_reads:     number of aligned reads to gene_id after post-alignment filter
mean_coverage:    average read depth of gene_id based on mapped_reads (total_gene_depth /
↪ covered_bases)
fraction_covered: proportion of the gene_id covered by at least one read (covered_bases /
↪ gene_length)
copy_number:      estimated copy number of gene_id based on mapped_reads (mean_coverage /
↪ median_marker_coverage)
```

Across-Samples

Merging across samples produces several outputs.

CNV results merged across samples are written to `midas2_output/merge/genes/genes_summary.tsv`

sample_name	*species_id*	*pangenome_size*	*covered_genes*	*fraction_covered*	*mean_coverage*	*aligned_reads*	*mapped_reads*	*marker_coverage*
sample1	100122	29165	2535	0.087	4.723	263395	53006	1.435
sample2	100122	9165	3212	0.110	16.095	1447684	263878	10.713

Besides `sample_name`, which indexes the entries, the other columns (`pangenome_size` through `marker_coverage`) are the same as in the per-sample genes summary output.

For each species, a matrix of gene-by-sample copy-number estimates—here species 102506 (*E. coli*)—are written to `midas2_output/merge/genes/102506.genes_copynum.tsv.lz4`.

gene_id	*sample1*	*sample2*
UHGG000587_00401	33.969154	19.891455
UHGG000587_01162	5.703398	2.821237
UHGG000587_00962	2.370930	0.289325

Similarly, a presence absence matrix is written to `midas2_output/merge/genes/102506.genes_preabs.tsv.lz4`.

gene_id	*sample1*	*sample2*
UHGG000587_00401	1	1
UHGG000587_01162	1	1
UHGG000587_00962	1	0

Raw vertical coverage data is reported in the same matrix form in `midas2_output/merge/genes/102506.genes_depth.tsv.lz4`.

gene_id	*sample1*	*sample2*
UHGG000587_00401	48.747945	213.090622
UHGG000587_01162	8.184746	30.222978
UHGG000587_00962	3.402439	3.099448

1.6.4 Advanced CNV Calling

Single-Sample Post-alignment Filter

Users can adjust post-alignment quality filter parameters via the command-line options (default vlaues indicated):

- `--mapq >= 2`: reads aligned to more than one genomic locations equally well are discarded (MAPQ=0,1)
- `--mapid >= 0.94`: discard read alignment with alignment identity < 0.94
- `--aln_readq >= 20`: discard read alignment with mean quality < 20
- `--aln_cov >= 0.75`: discard read alignment with alignment coverage < 0.75

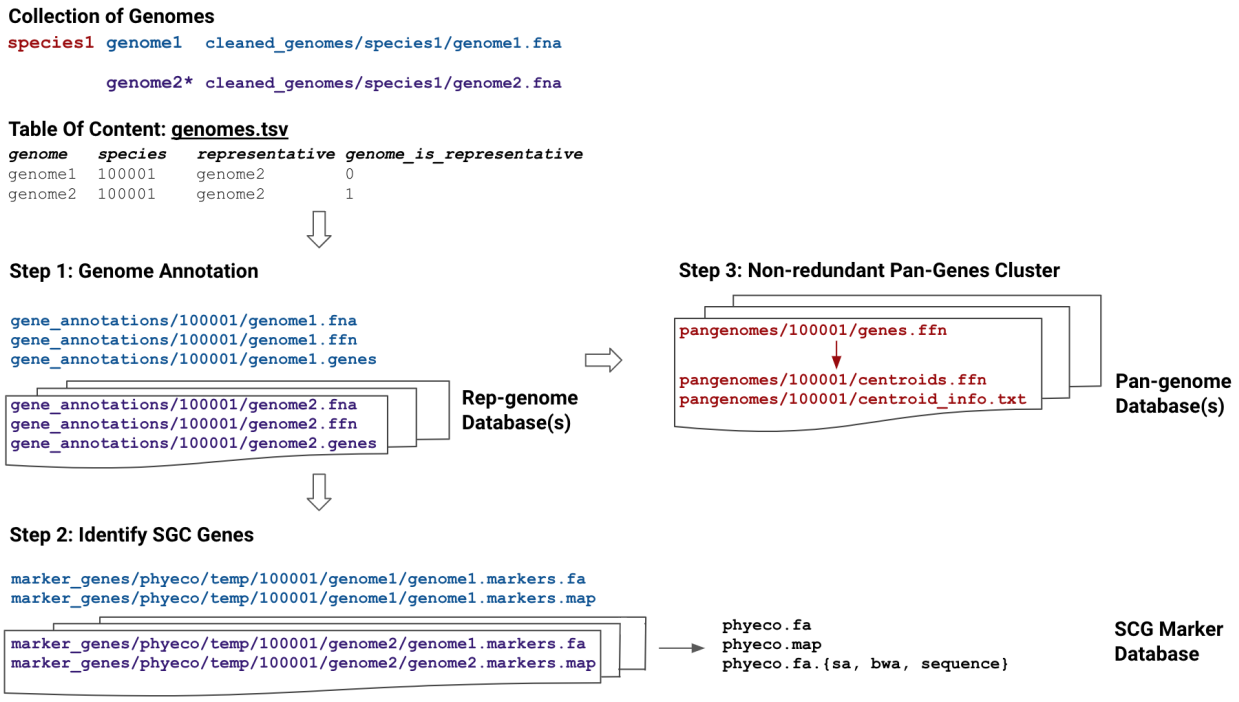
Adjust Population CNV Filters

The default `merge_genes` results are reported for pan-genes clustered at 95% identity (`cluster_pid`). It further quantify the presence/absence for pan-genes by comparing the `copy_number` with the user-defined minimal gene copy number (`min_copy`). `cluster_pid` and `min_copy` can be customized with the following command-line options:

- `--genome_depth`: filter out species with `mean_coverage` < 1X.
- `--min_copy`: genes with `copy_number` >= 0.35 are classified as present.
- `--cluster_pid`: gene CNV results can be reported at various clustering cutoffs {75, 80, 85, 90, 95, 99}.

1.7 Downloading the MIDASDB

MIDAS Reference Database (MIDASDB) is comprised of three components: representative genomes, species pangenomes, and marker genes. For each MIDASDB, six-digit numeric species ids are randomly assigned and stored in the corresponding metadata file (`metadata.tsv`).



For MIDAS2, we have already built two MIDASDBs from large, public, microbial genome databases:

```
midas2 database --list

uhgg 286997 genomes from 4644 species version 1.0
gtdb 258405 genomes from 47893 species version r202
```

For the purposes of this documentation we'll generally assume that we're working with the prebuilt `uhgg` MIDASDB and that the local mirror is in a subdirectory `my_midasdb_uhgg`.

Automatic database downloading is built into MIDAS2 analysis commands (e.g., `run_snps` and `run_genes`). Specifically, MIDAS2 will download a fraction of the full database; this subset is determined by which species are identified to be at high coverage.

However, when parallelizing computation across samples multiple commands might try to download the same database components simultaneously, a race condition. This may be problematic.

We therefore suggest that, for large-scale analyses, users pre-download the MIDASDB.

Users should start by downloading the taxonomic marker genes.

```
midas2 database \
    --init \
    --midasdb_name uhgg \
    --midasdb_dir my_midasdb_uhgg
```

This is everything needed to run *abundant species detection*.

It is possible to download an entire MIDASDB using the following command:

```
midas2 database \
  --download \
  --midasdb_name uhgg \
  --midasdb_dir my_midasdb_uhgg \
  --species all
```

This requires a large amount of data transfer and storage: 93 GB for `MIDASDB-uhgg` and 539 GB for `MIDASDB-gtdb`.

Note: The database would be much larger except that files are compressed with LZ4 to minimize storage requirements.

Alternatively, we strongly recommend that users take a **more customized approach to database loading**, taking advantage of species-level database sharding to download and decompress only the necessary portions of a MIDASDB.

Afterwards, we can collect a list of species present in a list of samples. Parsing the MIDAS2 *output files* (`midas2_output/merge/species/species_prevalence.tsv`) presents a convenient way to do this.

```
awk '$6 > 1 {print $1}' midas2_output/merge/species/species_prevalence.tsv > all_species_
↪list.tsv
```

Finally, we can download database components (both reference genomes and pangenome collections) based on these species.

```
midas2 database \
    --download \
    --midasdb_name uhgg \
    --midasdb_dir my_midasdb_uhgg \
    --species_list all_species_list.tsv
```

Afterwards, the single-sample parts of the SNV and CNV modules can be run in parallel and without a potential race condition.

Note: It is also possible for advance users to *contruct their own MIDASDB* from a custom genome collection (e.g. for metagenome assembled genomes).

1.8 Output Files and Directory Layout

In this section we refer to the output directory for single-sample as `midas_output/sample_name`, and across-samples as `midas_output`. Input command-line options can be found at *this page*.

1.8.1 Single-Sample Results Layout

MIDAS2 analysis usually starts with species selection which selects sufficiently abundant species in each sample (command `run_species`). After completing this step, users can run the SNV module with `run_snps` or the CNV module with `run_genes`.

Here is an example of output from a single-sample analysis in whch the species, SNV, and CNV modules were run, as it would appear in the local filesystem.

```
Output                                    Producer        Meaning
-------------------------------------------------------------------------------------------
↪--
{midas_output}/{sample_name}
  |- species
     |- species_profile.tsv               run_species     Summary of species coverage
     |- markers_profile.tsv               run_species     Per species marker coverage
  |- snps
     |- snps_summary.tsv                  run_snps        Summary of reads mapping to rep-
↪genome
     |- {species}.snps.tsv.lz4            run_snps        Per species pileups
  |- genes
     |- genes_summary.tsv                 run_genes       Summary of reads mapping to pan-
↪genome
     |- {species}.genes.tsv.lz4           run_genes       Per species pan-gene coverage

 |- temp
     |- snps
        |- repgenomes.bam                 run_snps        Rep-genome alignment file
        |- {species}/snps_XX.tsv.lz4
     |- genes
        |- pangenome.bam                  run_genes       Pan-genome alignment file
        |- {species}/genes_XX.tsv.lz4
  |- bt2_indexes
     |- snps/repgenomes.*                 run_snps        Sample-specific rep-genome␣
↪database
     |- genes/pangenomes.*                run_genes       Sample-specific pan-genome␣
↪database
```

1.8.2 Across-Samples Results Layout

For a collection of samples, population SNVs and CNVs can be estimated using the subcommands `merge_snps` and `merge_genes`. This is what the output looks like in the local filesystem.

```
Output                                                  Producer        Meaning
-------------------------------------------------------------------------------------------
↪----------------------
{midas_output}
  |- species
    |- species_prevalence.tsv                           merge_species   Per species summary␣
↪statistics across samples
    |- species/species_read_counts.tsv                  merge_species   Species-by-sample␣
↪reads counts matrix
    |- species/species_coverage.tsv                     merge_species   Species-by-sample␣
↪marker coverage matrix
    |- species/species_rel_abundance.tsv                merge_species   Species-by-sample␣
↪relative abundance matrix
  |- snps
    |- snps_summary.tsv                                 merge_snps      Alignment summary␣
↪statistics per sample
    |- {species}/{species}.snps_info.tsv.lz4            merge_snps      Per species SNV␣
↪metadata
    |- {species}/{species}.snps_freqs.tsv.lz4           merge_snps      Per species site-by-
↪sample MAF matrix
    |- {species}/{species}.snps_depth.tsv.lz4           merge_snps      Per species site-by-
↪sample reads depth matrix
  |-genes
    |- genes_summary.tsv                                merge_genes     Alignment summary␣
↪statistics per sample
    |- {species}/{species}.genes_presabs.tsv.lz4        merge_genes     Per species gene-by-
↪sample pre-abs matrix
    |- {species}/{species}.genes_copynum.tsv.lz4        merge_genes     Per species gene-by-
↪sample copy number matrix
    |- {species}/{species}.genes_depth.tsv.lz4          merge_genes     Per species gene-by-
↪sample reads depth matrix
```

1.8.3 MIDAS Reference Database Layout

To meet the challenge of increasing numbers of available genome sequences, MIDAS2 implemented a new database infrastructure, geared to run on AWS Batch and S3, to achieve elastic scaling for building MIDAS2 reference databases.

To be specific, the MIDAS2 reference database construction step can be executed in AWS using hundreds of r5d.24xlarge instances over a period of a couple of days, depositing built products in S3. For example, it took ~$80,000 and a week to build the species pan-genome for all 47,894 species of GTDB r202.

Table of Content

The new database infrastructure reads in a table of contents (TOC) file, containing genome-to-species assignment and a choice of representative genome for each species cluster. One TOC file (`genomes.tsv`) per MIDAS2 reference database. The TOC file has four columns, among which `genome_is_representative` specify whether the `genome` is the representative genome for the corresponding `species`. Only one `representative` per `species`.

genome	species	representative	genome_is_representative
GUT_GENOME138501	104351	GUT_GENOME269084	0
GUT_GENOME269084	104351	GUT_GENOME269084	1

By default, MIDAS2 inherits the representative genome assignments from published prokaryotic genome databases. Inspired by the importance of selecting proper reference genome for accurate SNV calling, this new infrastructure empowers users to dynamically re-assign the representative genomes, simply by modifying the `genomes.tsv` file accordingly.

Microbial Genome Collections

Unified Human Gastrointestinal Genome (UHGG)

A collection of 286,997 genomes assembled from metagenomes, isolates and single cells from human stool samples has been clustered into 4,644 gut-only species in UHGG 1.0 catalogues. The collection of all the UHGG genomes were mirrored in a S3 bucket, which serves as the input to the database construction. Six-digit numeric species ids were arbitrarily assigned. Instead of species name, these `species_id` are used as species identifier in all the reports generated by MIDAS2.

Genome Taxonomy Database (GTDB)

GTDB R06-RS202 contains 45,555 bacterial and 2,339 archaeal species clusters spanning 258,406 genomes, released on April 27th, 2021. The genome members for each species cluster are specified in the sp_clusters_r202.tsv, upon which order six-digit numeric species ids are assigned. GTDB only provided the sequences of the representative genomes, and we downloaded all the genomes from NCBI genomes repository using genome_updater.

Target Layout and Construction

A MIDAS2 reference database (MIDASDB) consists of three primary parts: rep-genome database, pan-genome database, and universal single copy genes (SGC) marker database. The target layout of any MIDASDB follows the same relative structure, based on the root directory of the database. The following toy example demonstrates the major steps to construct the MIDASDB and the target layout using a collection of two genomes (`genome1` and `genome2`) from one species cluster `species1`.

TODO: insert image

Inputs

The input collection of genomes needs to be organized in the format `cleaned_genomes/<species>/<genome>/<genome>.fna`. And the table of contents file `genomes.tsv` needs to be generated accordingly, with randomly assigned six-digit `species_id`, to replace the species name. The `genome` name can be kept as it is.

genome	species	representative	genome_is_representative
genome1	100001	genome2	0
genome2	100001	genome2	1

Rep-Genome Database

Genome annotation is performed using Prokka, and the annotated genes are kept under the directory of `genes_annotations/<species>/<genome>`. The rep-genome database (for the SNV module) only includes the gene annotations and sequences for the representative genomes, as specified in the table of contents file.

```
gene_annotations/100001/genome2/genome2.fna.lz4
gene_annotations/100001/genome2/genome2.ffn.lz4
gene_annotations/100001/genome2/genome2.genes.lz4
```

SCG Marker Database

Marker genes are defined as universal, single-copy gene families. MIDAS2 uses a subset (15) of the PhyEco gene families. The pre-computed HMM model of this set of 15 single copy genes (SCGs) are available at:

```
s3://microbiome-pollardlab/uhgg_v1/marker_gene_models/phyeco/marker_genes.hmm.lz4
s3://microbiome-pollardlab/uhgg_v1/marker_gene_models/phyeco/marker_genes.mapping_
↪cutoffs.lz4
```

For each annotated genome, the homologs of 15 SCGs are identified with `hmmsearch`, as well as the mapping of gene id to corresponding marker gene id, under the directory of `marker_genes/phyeco/temp/<species>/<genome>`.

```
marker_genes/phyeco/temp/100001/genome2/genome2.markers.fa
marker_genes/phyeco/temp/100001/genome2/genome2.markers.map
```

For all the representative genomes, the identified marker genes are concatenated into monolithic `marker_genes.fa`, from which a `hs-blastn` index is constructed. The indexed `marker_genes.fa` serves as the SCG marker databases.

```
marker_genes/phyeco/marker_genes.fa
marker_genes/phyeco/marker_genes.fa.sa
marker_genes/phyeco/marker_genes.fa.bwt
marker_genes/phyeco/marker_genes.fa.sequence
```

Pan-Genome Database

Species-level pan-genome refers to the set of non-redundant genes that represent the genetic diversity within one species cluster.

In order to construct the pan-genome database for each species, the first step is to concatenate the annotated genes from all genomes within the species cluster into `pangenomes/100001/genes.ffn`. The second step, which is also the most time-consuming step, is to cluster the concatenated genes based on 99% percent identity (PID) using vsearch. Each cluster is represented by the gene at its center - centroid gene (`centroids.99.ffn`). The `centroid.99` genes are further on clustered to 95, 90, ..., PID, respectively, and the mapping relationships are listed in `centroid_info.txt`. The top level `centroids.ffn` file represents the 99 percent identity clusters, and it serves as the species pan-genome databases.

Reads are aligned to the pan-genome database to determine the gene content and average copy number of strains in a sample (`run_genes` command), and reads can optionally aggregated into gene clusters at any of the lower clustering thresholds across samples (`merge_genes` command).

```
pangenomes/100001/centroids.ffn
pangenomes/100001/centroid_info.txt
```

1.9 Advanced Topics

Contents

1.9.1 Population SNV Calling

In this section, we will the compute of population SNV, the chunkified implementation of pileup, as well as filters of species, samples and genomic sites in the **SNV** module.

Important Concepts

1. **<species, relevant samples> selection**

   Population SNV analysis restricts attention to "sufficiently well" covered species in "sufficiently many" samples.

   To be specific, a given <species, sample> pair will only be kept (by default) if it has more than 40% horizontal genome coverage (`genome_coverage`) and 5X vertical genome coverage (`genome_depth`). Furthermore, only "sufficiently prevalent" species with "sufficiently many" samples (`sample_counts`) would be included for the population SNV analysis. Therefore, different species may have different lists of relevant samples.

2. **<site, relevant samples> selection**

   For each genomic site, a sample is considered to be "relevant" if the corresponding site depth falls between the range defined by the input arguments `site_depth` and `` `site_ratio * mean_genome_coverage ``; otherwise it is ignored for the across-samples SNV compute.

   Therefore, different genomic sites from the same species may have different panels of "relevant samples". Genomic site prevalence can be computed as the ratio of the number of relevant samples for the given site over the total number of relevant samples for the given species.

3. **relevant site**

   For each species, a site is considered to be "relevant" if the site prevalence meets the range defined by the input arguments `snv_type` and `` `site_prev ``. By default, common SNV with more than 90% prevalence are reported.

Population SNV Computation

There are three main steps to compute and report population SNVs in MIDAS2.

First, for each relevant genomic site, MIDAS2 determines the set of alleles present across all relevant samples. Specifically, for each allele (A, C, G, T), `merge_snps` command

1. tallys the sample counts (`sc_`) of relevant samples containing corresponding allele (`scA:scT`)
2. sums up the read counts (`` rc`_`) of the corresponding allele across all the relevant samples (``rc_G:rc_T``).

site_id	rc_A	rc_C	rc_G	rc_T	sc_A	sc_C	sc_G	sc_T
gnl|Prokka|UHGG000587_14|34360|A	26	10	0	0	1	2	0	0

Second, population major and minor alleles for a single site can be computed based on the accumulated read counts or sample counts across all relevant samples. The population major allele refers to the most abundant (read count) or prevalent (sample count) allele, and the population minor allele refers to the second most abundant or prevalent allele.

For example, the population major allele of the site `gnl|Prokka|UHGG000587_14|34360|A` in the above example is `A` defined by accumulated read counts and `C` defined by accumulated sample counts.

Third, MIDAS2 collects and reports the sample-by-site matrix of the corresponding (1) site depth and (2) allele frequency of the above calculated population minor allele for all the relevant samples. In these two matrices, MIDAS2 encode `site_depth = 0` and `allele_frequency = -1` with the special meaning of missing <site, sample> pair.

Chunkified Pileup Implementation

Both single-sample and across-samples pileup are parallelized on the unit of chunk of sites, which is indexed by <species_id, chunk_id>. When all chunks from the same species finish processing, then chunk-level pileup results will merged into species-level pileup result.

This implementation makes population SNV analysis across thousands of samples possible. To compute the population SNV for one chunk, all the pileup results of corresponding sites across all the samples need to be read into memory. With the uses of multiple CPUs, multiple chunks can be processed at the same time. Therefore, for large collections of samples, we recommend higher CPU counts and smaller chunk size to optimally balance memory and I/O usage, especially for highly prevalent species. Users can adjust the number of sites per chunk via `chunk_size` (default value = 1000000). MIDAS2 also has a `robust_chunk` option, where assigning different chunk sizes to different species based on the species prevalence.

1.9.2 Build Your Own MIDASDB

MIDAS2 users can locally build a new MIDASDB for a custom collection of genomes. The target layout of MIDASDB can be found at *MIDASDB Layout*. This section is focused specifically on the database construction commands.

Table of Content (TOC)

To start with, users need to organize the genomes in a specific format and produce the TOC `genomes.tsv` as described in the MIDASDB Layout.

We have prepared a toy collections of genomes at the `tests/genomes`, and we will build the new MIDASDB under the directory of `tests/genomes`. There are two command-line parameters that users need to pass:

- `--debug`: keep the local file after successfully build the database
- `--force`: re-build the database even if already locally exists

Six-digit numeric species ids are randomly assigned and can be stored in the corresponding metadata file (`metadata.tsv`)

MIDAS2 reserves `--midasdb_name newdb` for building a custom MIDASDB, and the new MIDASDB will be built at `--midasdb_dir`.

Rep-genome

First, annotate all the genomes:

```
midas2 annotate_genome --species all
  --midasdb_name newdb --midasdb_dir my_new_midasdb \
  --debug --force

midas2 build_midasdb --generate_gene_feature \
  --genomes all \
  --midasdb_name newdb --midasdb_dir my_new_midasdb
  --debug --force
```

SCG Markers

Second, infer SCGs for all the genomes and build marker database:

```
midas2 infer_markers --genomes all
  --midasdb_name newdb --midasdb_dir my_new_midasdb \
  --debug --force

midas2 build_midasdb --build_markerdb \
  --midasdb_name newdb --midasdb_dir my_new_midasdb \
  --debug --force
```

Pan-genome

Third, build species pangenomes:

```
midas2 build_pangenome --species all \
  --midasdb_name newdb --midasdb_dir my_new_midasdb \
  --debug --force

midas2 build_midasdb --generate_cluster_info \
  --species all \
  --midasdb_name newdb --midasdb_dir my_new_midasdb \
  --debug --force
```

1.9.3 Build Your Own Genome Index

MIDAS2 builds sample-specific rep-genome or pan-genome index for species in the restricted species profile. However, we recognize the need of using one comprehensive list of species across samples in the same study. In this section, we will go over the steps of building one genome index containing a customized list of species across a set of samples.

We presuppose users have already completed the *across-samples species profiling* and have `midas2_output/merge/species/species_prevalence.tsv` ready for the set of samples.

Species Selection

Users can select species based on the prevalence from the `species_prevalence.tsv` file, e.g. the list of species that are present in at least one sample, by customizing the `--select_by` and `--selectd_threshold` to the `build_bowtie2db` command.

Build Genome Index

In this section, we will keep using the *example data* from Quickstart.

```
midas2 build_bowtie2db \
  --midasdb_name uhgg --midasdb_dir my_midasdb_uhgg \
  --select_by sample_counts \
  --select_threshold 2 \
  --bt2_indexes_name repgenomes \
  --bt2_indexes_dir one_bt2_indexes \
  --num_cores 8
```

And users can locate the generated rep-genome database at `one_bt2_indexes/repgenomes`, and the list of species in the rep-genome is at `one_bt2_indexes/repgenomes.species`.

Use Prebuilt Genome Index

If taking this approach, for the single-sample SNV or CNV analysis, users can pass the pre-built rep-genome to `run_snps` analysis (pan-genome for `run_genes`), as follows:

```
midas2 run_snps
  --sample_name sample1 \
  -1 reads/sample1_R1.fastq.gz \
  --midasdb_name uhgg \
  --midasdb_dir my_midasdb_uhgg \
  --prebuilt_bowtie2_indexes one_bt2_indexes/repgenomes \
  --prebuilt_bowtie2_species one_bt2_indexes/repgenomes.species \
  --select_threshold=-1 \
  --num_cores 8 \
  ${midas_output}
```

1.9.4 Developer Notes

Export Your Conda Environment

```
conda update --all
conda clean -all
conda env export --no-builds | grep -v "^prefix:" > midas2.updated.yml
```

1.10 Example Questions

1.11 What percent of observed data can be accounted for by the major and minor alleles?

For a single sample, MIDAS2 reports the read counts for all four nucleotides (A, C, G, T) for all the genomic sites covered by more than two reads. For a collection of samples, MIDAS2 accumulated either the read counts or sample counts for all four nucleotides and reported the population major/minor allele, and sample-wise population major/minor allele frequency.

> **Warning:** MIDAS and MIDAS2 are not a consensus SNP calling tool.

To answer this question, we investigated one of the top prevalent species *Bacteroides_B dorei* using the PREDICT study. Among the 942 samples metagenotyped by MIDAS2, 354 samples were inferred with a single dominant strain of *Bacteroides_B dorei*, and 588 samples were inferred with multiple strains.

For all the genomic sites used for the the quasi-phasable (QP) compute, we looked at the distribution of the unexplained ratio of two types of major/minor alleles:

(1) single-sample major/minor allele(s),

(2) population major/minor alleles(s).

Note: It is worth noting that the unexplained ratio of a fixed or bi-allelic site is 0.

1.11.1 Results

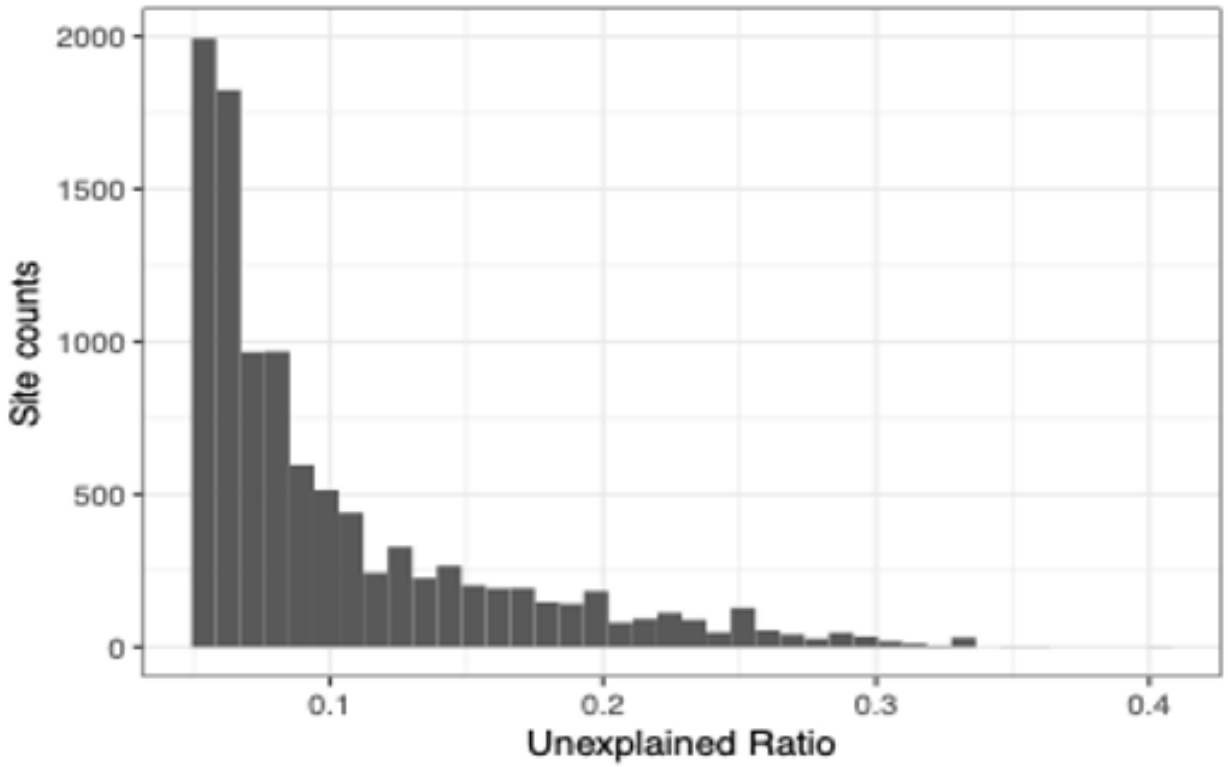
Single-Sample Major/Minor Alleles

For genomic sites genotyped with one or two alleles, the single-sample major/minor allele by definition explained 100% of the read counts.

First, we looked at the percentage of genotyped sites that have more than three alleles:

Sample Types	*Average Percentage of Sites with More Than Two Alleles*
Samples with a dominant strain of B. dorei	0.00609%
Samples with multiple strains of B. dorei	0.0342%

Second, we investigated the ratio of read counts unexplained by the most two abundant alleles (single-sample major and minor allele), for all the 10243 sites (from all 942 samples) genotyped with more than two alleles (*unexplained ratio*).



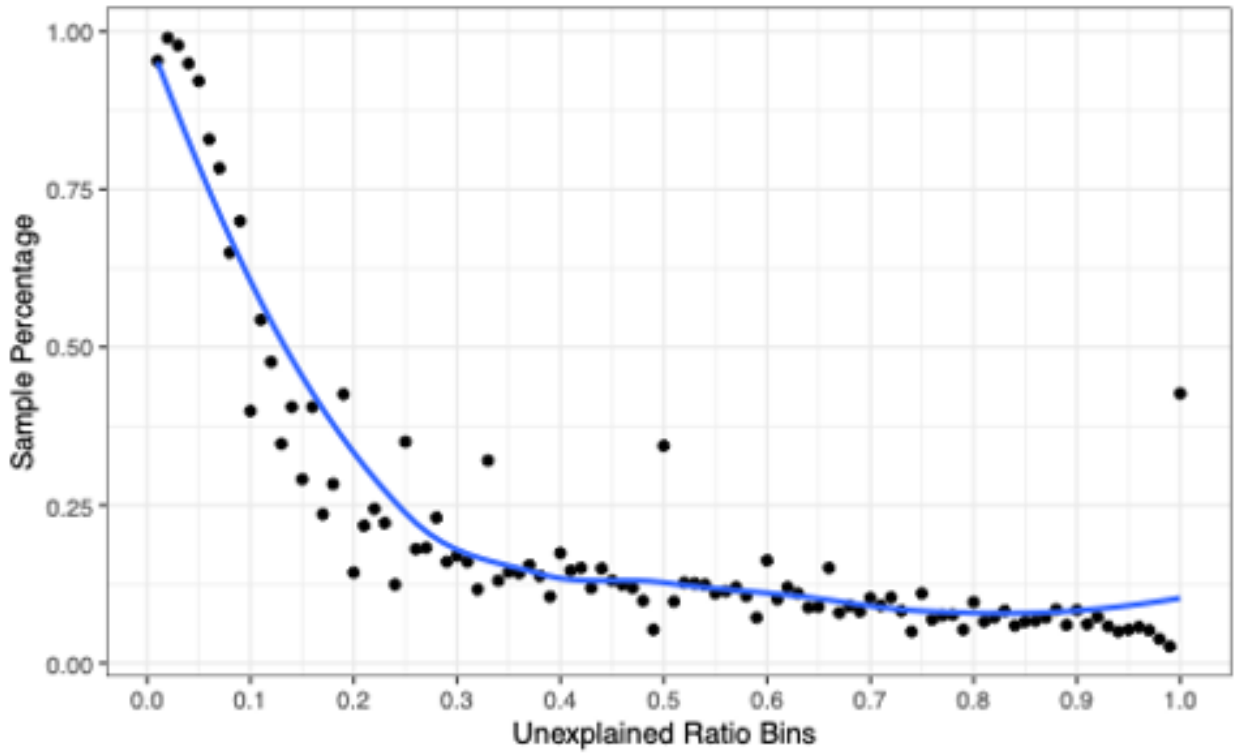
As shown in the histogram above, 65.25% of the 10243 sites were reported with unexplained ratio less than 10%, and 96.30% were reported with unexplained ratio less than 25%.

Population Major/Minor Alleles

Recall that for each genomic site, the population major allele was voted as the most prevalent allele based on accumulated sample counts, and population minor allele was voted as the second most prevalent allele in this analysis. MIDAS2 then reported the corresponding allele frequency of the population major/minor allele for each sample.

On average 99.3% of the genomic sites can be fully explained by the population major/minor alleles. We further computed the ratio of read counts unexplained by the population major/minor alleles for the rest 0.7% sites (per sample). We binned the unexplained ratio into 100 bins between 0.01 and 1.0.

We first looked at the distribution of the number of samples for the binned unexplained ratio.

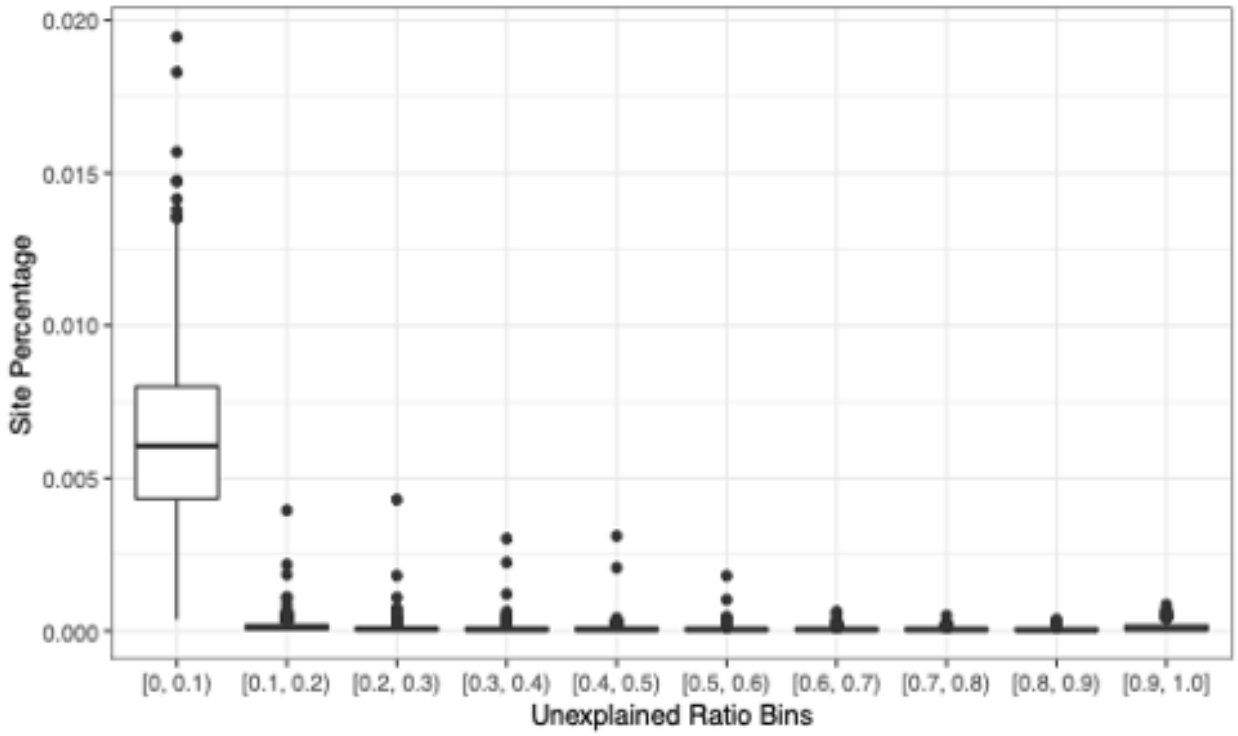


We then looked at the percentage of sites for the binned unexplained ratio for all the sites.

In summary, the unexplained ratio for most of the sites (not fully explained by the population major/minor alleles) is below 10% (the average per sample site percentage for the (0, 0.1) bin is 0.63%).

1.12 What is the detection power of MIDAS2 when multiple strains are present per species?

We designed a simple simulation experiment to investigate the question: how often can MIDAS2 detect the minor allele for a sample with two strains? What is the precision and recall of the genotyping results?



1.12.1 Simulation Design

We simulated reads from two NCBI strains of *Phocaeicola-dorei* (MIDASDB UHGG species_id 102478) with varied sequence depths, shown as following:

Mixture Strains	*NCBI Accession*	*NCBI Taxonomy*	*ANI to Rep-genome*	*Simulated Coverage*	*Between-strains ANI*
Strain1	GCF_000158335.2	Phocaeicola-dorei-5-1-36/D4	98.86%	20X	98.91%
Strain2	GCF.013009555.1	Phocaeicola-dorei-DSM-17855	98.94%	1X 5X 10X 15X 20X	98.91%

- Rep-genome refers to the default representative genome assigned by UHGG database (GUT_GENOME143505).

1.12.2 Bioinformatics

We simulated five samples composed of the above mentioned two strains at varying mixing ratios. MIDAS2 single-sample SNV analysis with the default parameters for reads alignment and post-alignment filtering were run. With the single-sample genotyping results, two additional filters were applied: `site depth >= 5X` and `allele frequency >= 0.05`.

Lower sequencing coverage (e.g. 0.1X) really means the horizontal coverage is lower. However in terms of the vertical coverage for each site, the above filter required a minimal 5X site depth, and minimal 2X to call an allele (default filter by MIDA2). Therefore the lowest sequence coverage we simulated here is 1X.

Warning: MIDAS2 doesn't have the underlying assumption that a sample will have a single dominant strain per species.

1.12.3 Analysis

We determined the correct genotypes for the two mixture strains by using whole-genome alignments via *nucmer* (true_sites). Given the high within-species ANI between the two mixture strains, there are 16603 ALT sites and 4193849 REF sites, where ALT refers to genomic sites where two strains disagree with each other, and REF site refers to the site that agrees with each other. Here the modified population SNVs can be interpreted as genotyping errors in this context.

For each genotyped genomic site, we computed a modified version of population SNVs introduced by inStrain: a population SNV refers to a site where either the major allele and/or the minor allele differ from the two strains. Furtheron, we computed the precision and recall of all the genotyped sites:

- Precision: number of correctly genotyped sites / total number of genotype-able sites
- Recall: number of correctly genotyped sites / total number of sites in true_sites

1.12.4 Results

Strain1	*Strain2*	*Strain Ratio*	*Type*	*Population SNVs*	*Precision*	*Recall*
20X	1X	0.05	ALT	3	0.003	0.003
20X	1X	0.05	REF	47	1	0.987
20X	5X	0.25	ALT	0	0.779	0.758
20X	5X	0.25	REF	27	1	0.992
20X	10X	0.5	ALT	0	0.839	0.833
20X	10X	0.5	REF	22	1	0.993
20X	15X	0.75	ALT	0	0.855.03	0.851
20X	15X	0.75	REF	23	1	0.994
20X	20X	1	ALT	0	0.863	0.859
20X	20X	1	REF	21	1	0.994

For a sample with a mixture of two strains with average vertical coverage higher than 5X, both the precision and recall of the genotyped ALT sites increased with the genome vertical coverage. Since we applied the minimal site depth = 5X filter, most sites for the 1X sample were not detected.

Strain deconvolution is out of scope of MIDAS2, yet the results of the MIDAS2 SNV module can be used as the input for strain deconvolution tools, such as StrainFacts.

1.13 Glossary

- **Module**: An major component of MIDAS2, including Species, SNVs, CNVs and MIDASDB building.
- **Analysis**: Either the single-sample alignment-based tallying of reads mapped to SCGs (species), a representative genome (SNVs), or genes in a pangenome (CNVs) depending on which module is run OR the merging, filtering, and summarization of these results across samples.
- **Workflow**: The overall conceptual order of events from an implicit database and pre-requisite shotgun metagenomic samples through species selection and SNPs/genes modules, to results that will go into some sort of downstream analysis.
- **Reference database**: The upstream UHGG, GTDB, something user supplies, etc. that is then pre-processed to make a specific…
- **MIDASDB**: The pre-processed reference data in a format to be consumed by the MIDAS2 modules, including marker genes, reference genomes, pangenomes, etc.

- **genome Index**: Rather than bowtie2 database or some other ambiguous term
- **Species-level pangenome**: refers to the set of non-redundant genes (centroids) clustered from all the genomes within one species cluster.

1.14 Acknowledgements

MIDAS2 was developed by Chunyu Zhao and Boris Dimitrov in the Pollard Lab at Chan Zuckerberg Biohub.

Special thanks to Byron J. Smith for tremendous help with this documentation.

More to come :)