
DTU

Patrick Munk

Single & co-abundance binning: metabat2, vamb

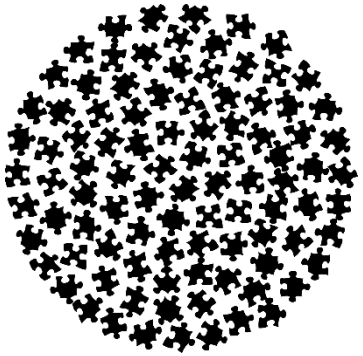
Metagenomic challenge

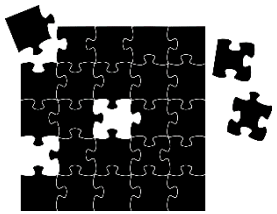
- Easy dataset!
- Only 32000 puzzle pieces!
- They all fit together in a single puzzle

Happy ph.d. student

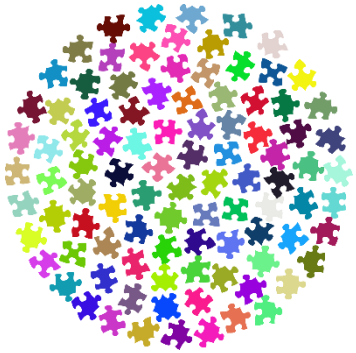
- Imagine 15 x 10^6 pieces from 480 different puzzles
- Now mix them all
- THEN solve them

Genome assembly





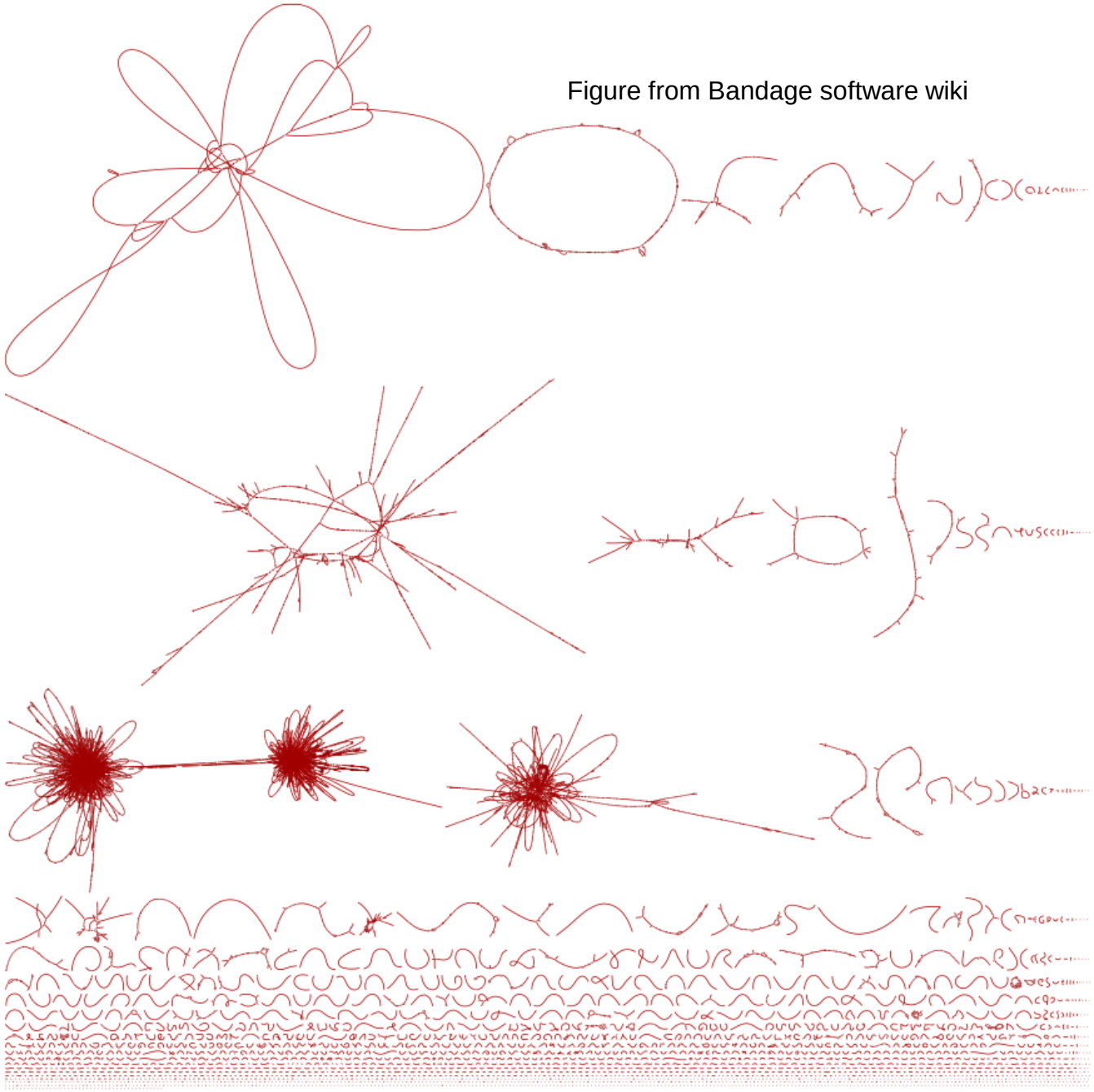
Metagenomic assembly



Contig

Draft genome

Figure from Bandage software wiki

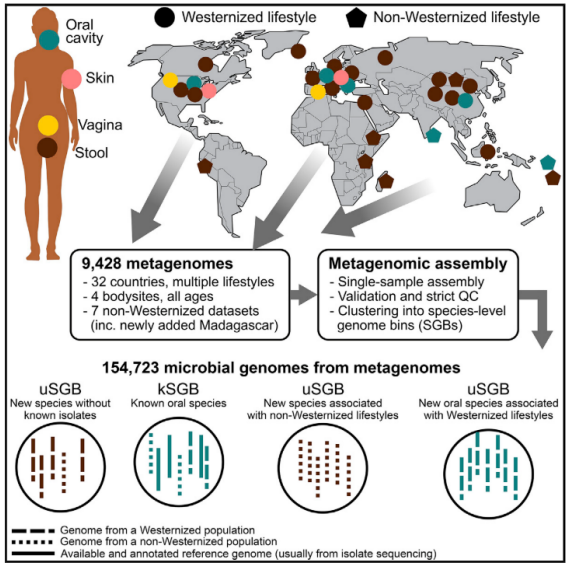


Cell

Resource

Extensive Unexplored Human Microbiome Diversity Revealed by Over 150,000 Genomes from Metagenomes Spanning Age, Geography, and Lifestyle

Graphical Abstract



Highlights

- Large-scale metagenomic assembly uncovered thousands of new human microbiome species
- The new genome resource increases the mappability of gut metagenomes over 87%
- Some of the newly discovered species comprise thousands of reconstructed genomes
- Non-Westernized populations harbor a large fraction of the newly discovered species

Authors

Edoardo Pasolli, Francesco Asnicar, Serena Manara, ..., Christopher Quince, Curtis Huttenhower, Nicola Segata

Correspondence

nicola.segata@unitn.it

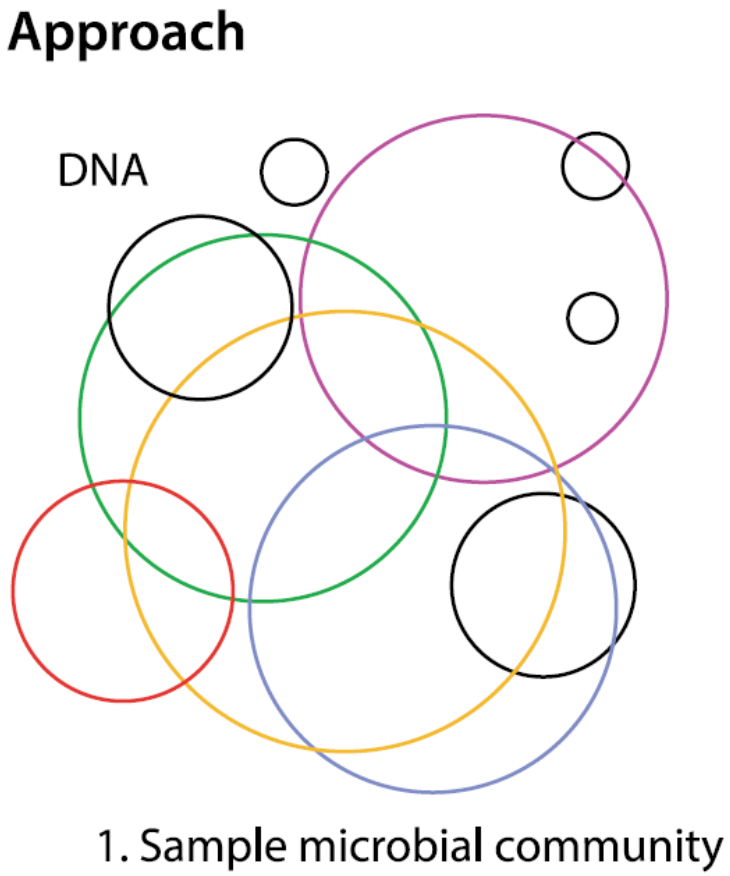
In Brief

The human microbiome harbors many unidentified species. By large-scale metagenomic assembly of samples from diverse populations, we uncovered >150,000 microbial genomes that are recapitulated in 4,930 species. Many species (77%) were never described before, increase the mappability of metagenomes, and expand our understanding of global body-wide human microbiomes.

Pasolli et al., 2019, Cell *176*, 649–662
January 24, 2019
https://doi.org/10.1016/j.cell.2019.01.001

CellPress

What is genome binning?



Al-shayeb, B et al. (2020). Nature. March 2019. https://doi.org/10.1038/s41586-020-2007-4

Question

How would YOU try to determine which contigs belong together?

Genome binning : tools and strategies

- **Automated tools**
  - MetaBat2 (DA/NC)
  - groopM (DA/NC)
  - CONCOCT (DA/NC)
  - MaxBin2 (DA/NC)
  - Canopy (DA)
  - ABAWACA
  - VAMB (DA/NC)

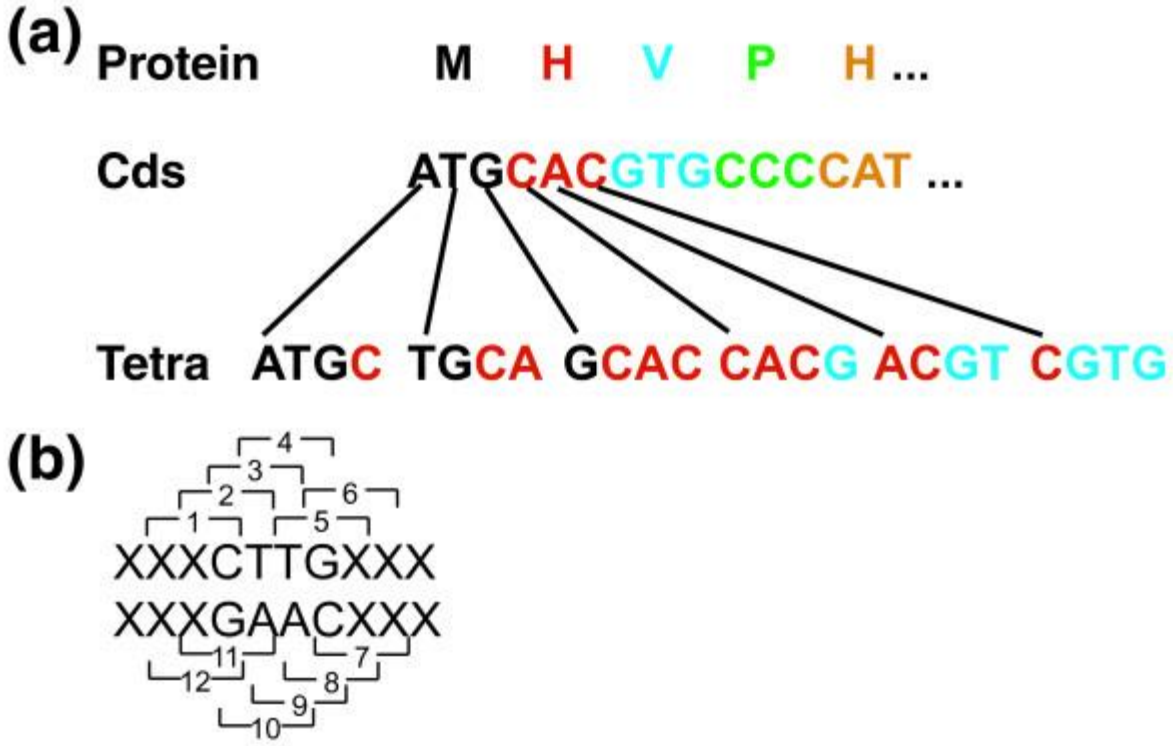
- **Manual intervention**
  - Emergent Self-Organizing Maps (ESOMs, NC)
  - VizBin (NC)
  - Custom solutions, e.g. ggplot, ggKbase (DA/NC)

Differential abundance (DA)

Nucleotide composition (NC)

Overview : abundance-based binning

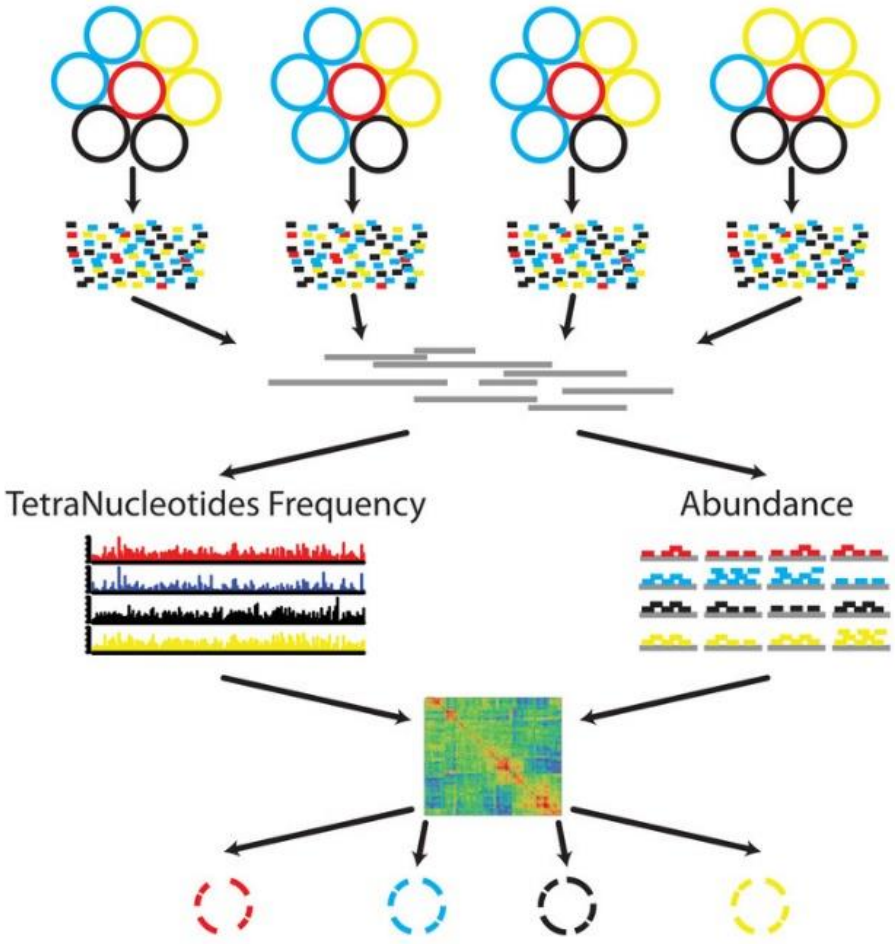
Overview : Nucleotide composition binning



Gregory Dick et al. (2009) Genome Biology

MetaBat: Hybrid DA / NC binning

- Uses both tetranucleotide frequency and abundance to bin contigs
- Works wells with both single and multiple libraries



Preprocessing

1. Samples from multiple sites or times
2. Metagenome libraries
3. Initial de-novo assembly using the combined library

MetaBAT

4. Calculate TNF for each contig
5. Calculate Abundance per library for each contig
6. Calculate the pairwise distance matrix using pre-trained probabilistic models
7. Forming genome bins iteratively

Kang DD et al. (2015)

Exercise today: Genome bin your MG assembly

Four overall steps in your genome binning

1. Quantify all your metagenomic scaffolds > 1kbp in your assembly (BBmap)
2. Make a sorted, indexed BAM file from the mapping reads (Samtools)
3. Make a coverage table from your BAM file (jgi_summarize_bam_contig_depths)
4. Bin draft genomes from your assembly (MetaBat2)
5. Try co-abundance binning (coverage table from multiple BAMs)

 # Modules needed

```
module load ngs tools
module load perl/5.24.0
module load metabat/2.12.1
module load samtools/1.9
module load java/1.8.0
module load bbmap/36.49
```

Commands to run

1. Map your trimmed reads against the 1kbp+ metaspades scaffolds

mkdir -p mapped/

bbmap.sh in=DTU_2017_1011718_1_MG_PT_PO_R1.fastq.gz in2=DTU_2017_1011718_1_MG_PT_PO_R2.fastq.gz minid=0.90 threads=20 ref=scaffolds_gt1000bp.fa outm=mapped/mapped.sam overwrite=t nodisk=t

2. Make a sorted, indexed BAM from the output SAM

samtools view -bSh1 mapped/mapped.sam | samtools sort -m 20G -@ 3 > mapped/mapped.sort.bam

3. Run the JGI script that converts one or more BAMs to a coverage table

jgi_summarize_bam_contig_depths mapped/mapped.sort.bam --outputDepth mapped/coverage.txt

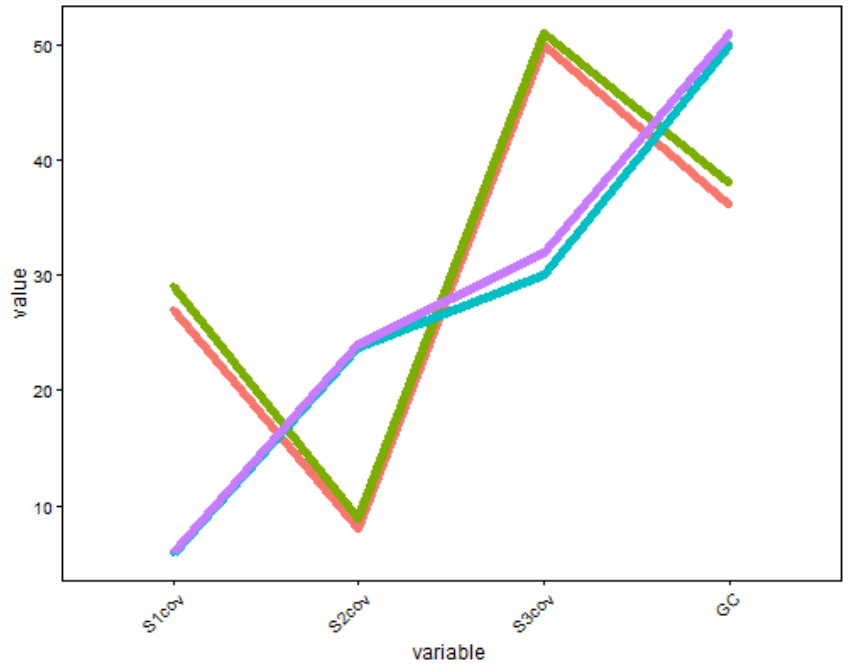
4. Run metabat2 to bin the 1kbp+ scaffolds into draft genomes using the coverage table

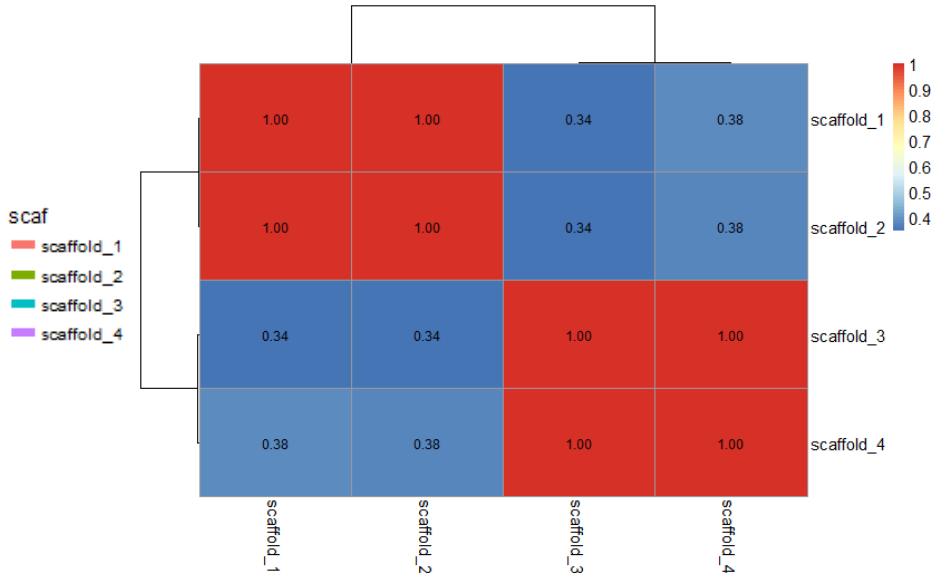
mkdir -p metabat2/

metabat -t 20 -v -m 1500 --saveCls -i scaffolds_gt1000bp.fa -a mapped/coverage.txt -o metabat2/scaffolds_gt1000bp.bin

Combining DA and NC information

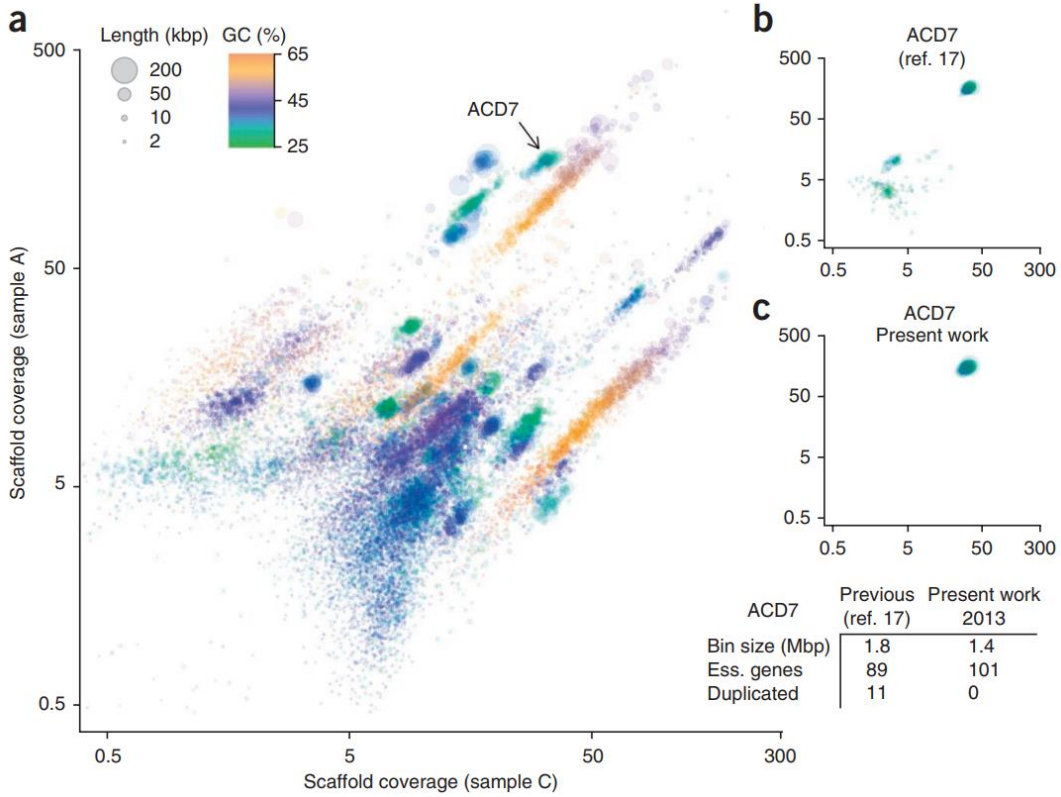
	Sample1 cov	Sample 2 cov	Sample 3 cov	GC %
scaffold_1	27	8	50	36
scaffold_2	29	9	51	38
scaffold_3	5,8	23,8	30	50
scaffold_4	6	24	32	51





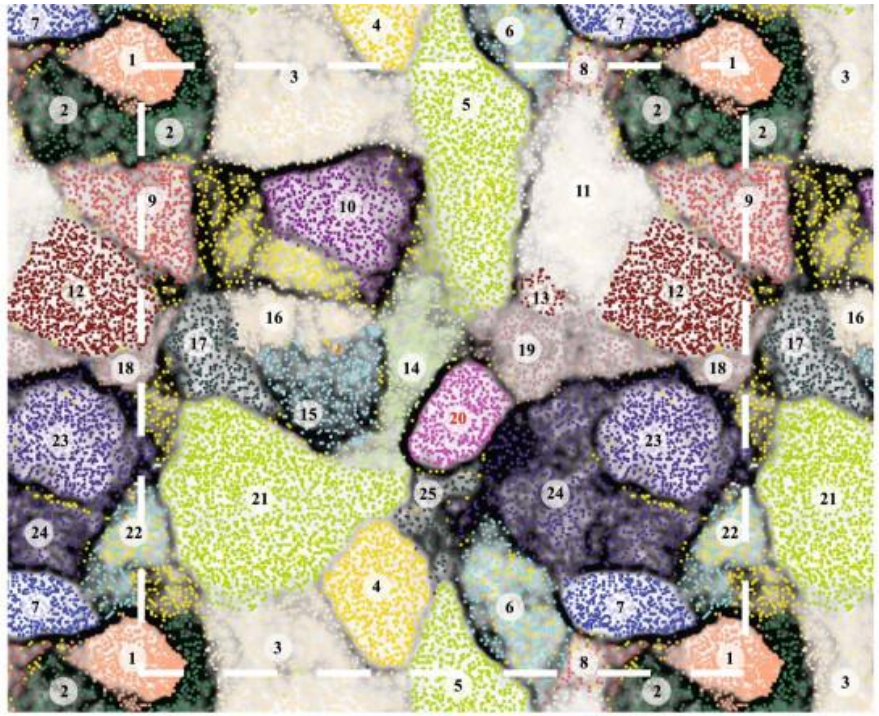
Genome binning based on extraction differences

- Using different DNA extraction protocols on the same sample, can inform binning
- If one protocol is more efficient at extraction DNA from certain species, we expect all contigs from said species to be over



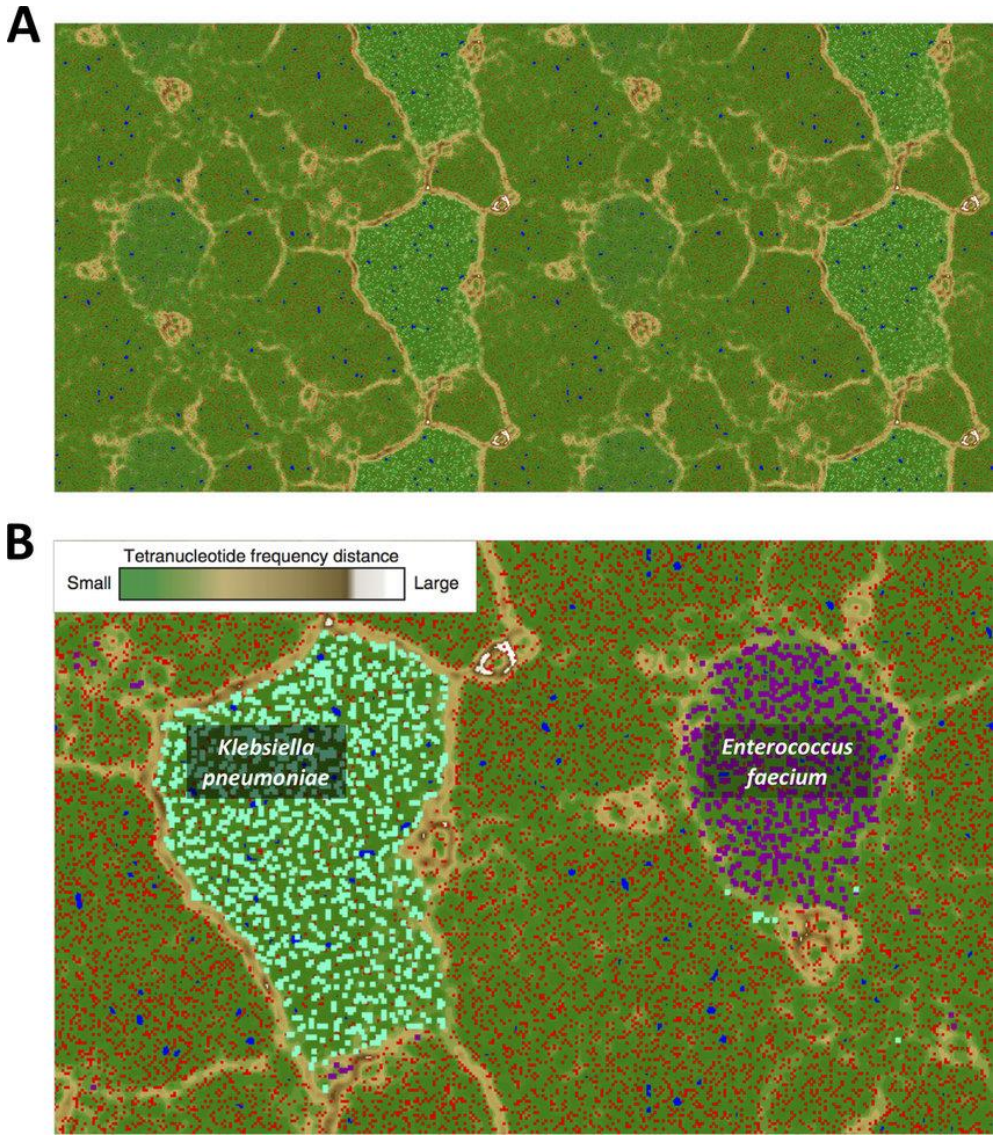
Albertsen M et al. (2013)

ESOMs



Unbinned		12, 13	*Actinomyces urogenitalis*
Clostridium butyricum	1	12	*Actinomyces urogenitalis - Phage*
Leuconostoc sp.	2	14	*Escherichia coli - Strain B*
Veillonella sp. - Species A	3, 16	14	*Enterococcus faecalis - Plasmid*
Enterococcus faecalis	4	14	*Escherichia coli - Strain A/B - Plasmid*
Escherichia coli - Strain A	5, 21	15	*Propionibacterium sp.*
Escherichia coli - Strain A - Phage	5	16	*Veillonella sp. - Species A - Plasmid*
Staphylococcus sp.	6, 22	22	*Veillonella sp. - Species A - Phage*
Staphylococcus sp. - Plasmid A	unbinned	17, 25	*Negativicoccus succinicivorans*
Staphylococcus sp. - Plasmid B	unbinned	18, 19	*Veillonella sp. - Species B*
Staphylococcus sp. - Phage	unbinned	23	*Veillonella sp. - Species B - Phage*
Streptococcus anginosus	7	20	*Varibaculum cambriense*
Clostridium bartlettii	8, 9	23, 24	*Streptococcus parasanguinis*
Streptococcus sp.	10	23	*Streptococcus parasanguinis - Plasmid*
Veillonella dispar	11		

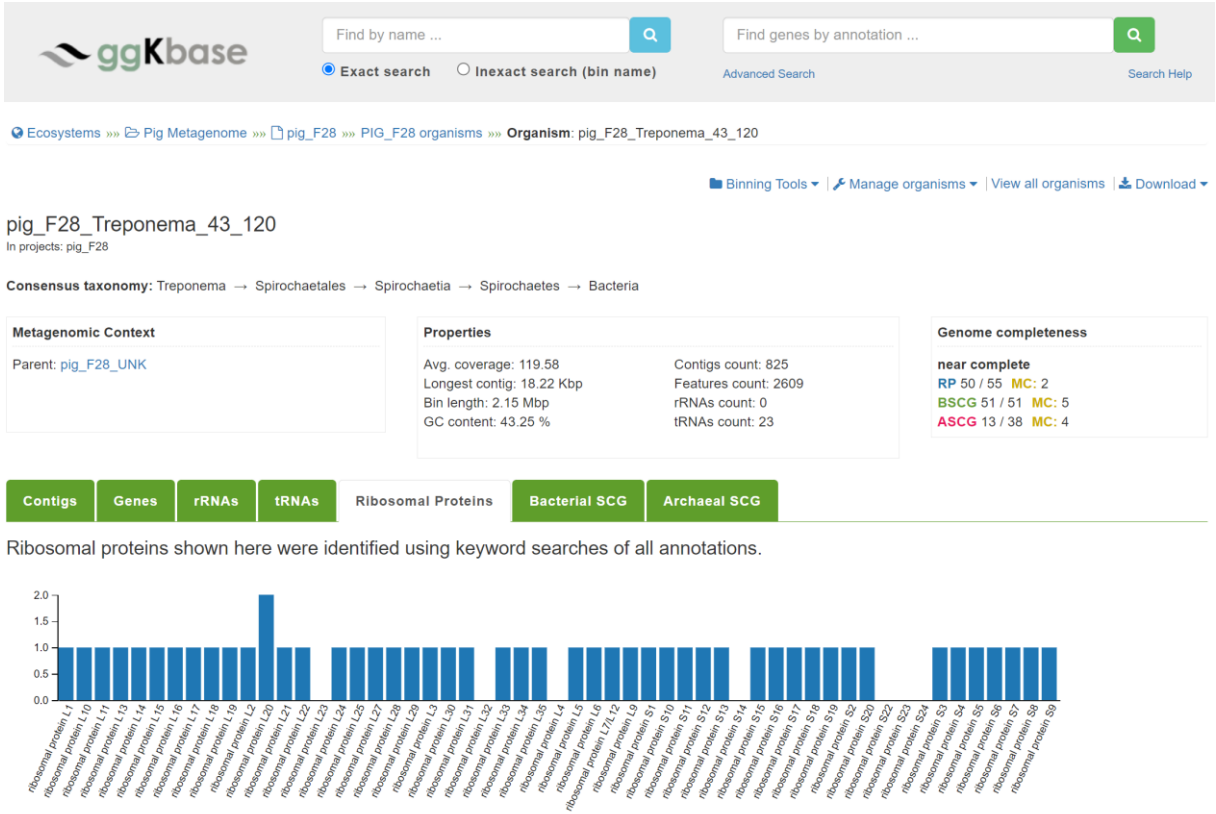
Brown, Christopher et al. (2013). Microbiome. 1. 30. 10.1186/2049-2618-1-30.



Mu, Andre et al (2019). mSphere

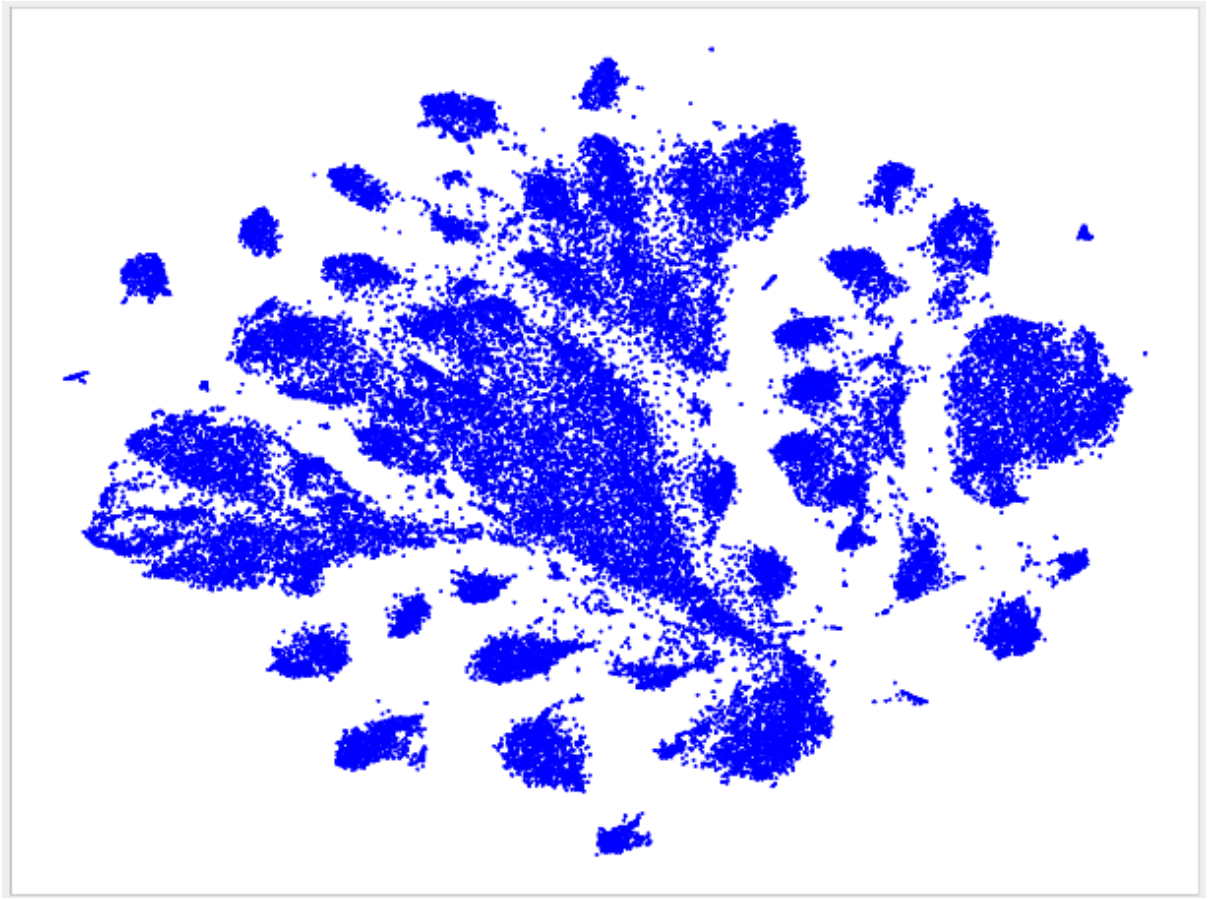
ggKbase visual binning

- Combination of %GC and read depth
- Sanity check bins in terms of single-copy genes and/or ribosomal proteins

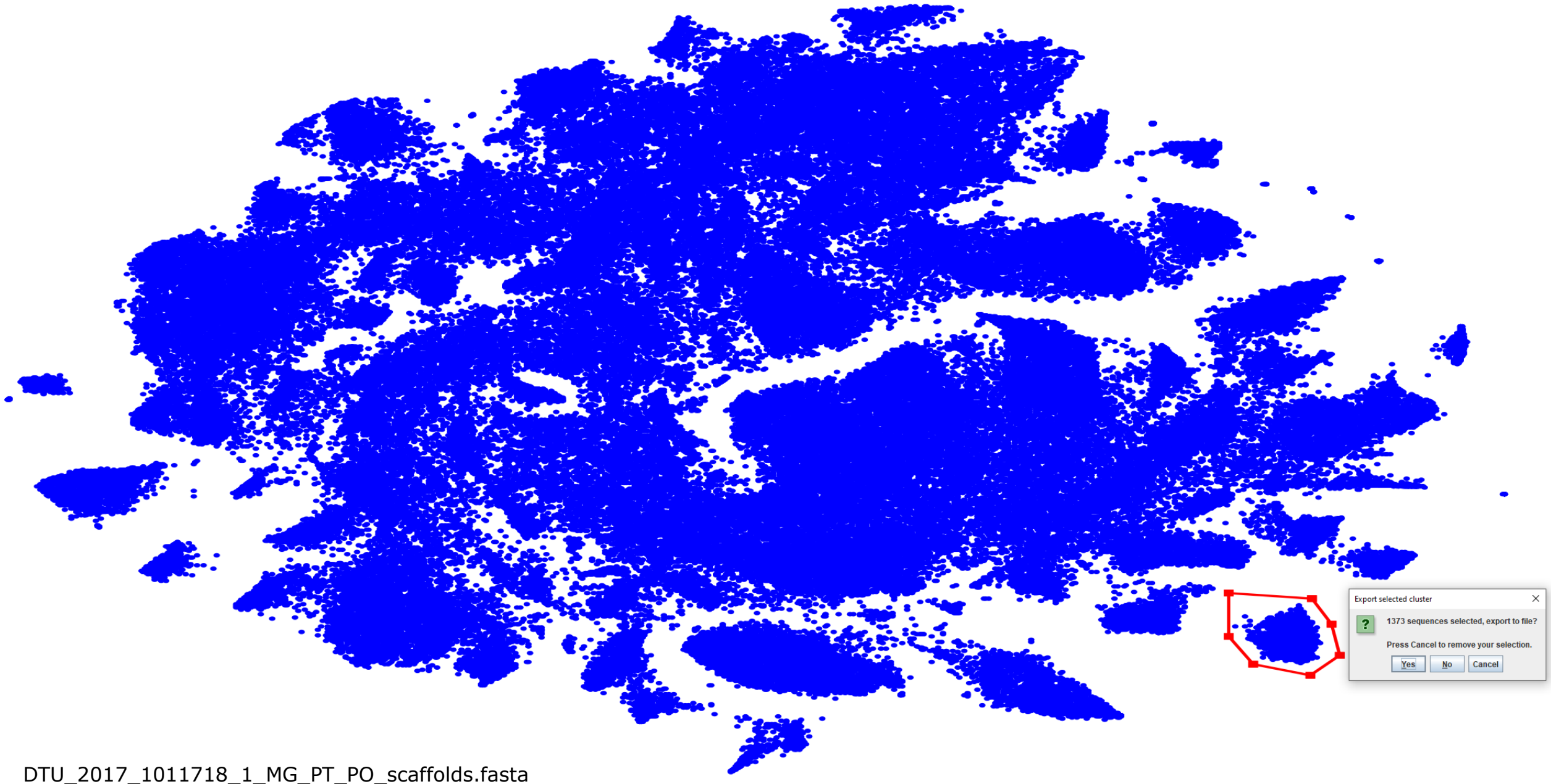


VizBin

- Barnes-Hut t-SNE clustering of contigs based on kmer composition
- Gate-based selection from visual inspection



VizBin (claczny.github.io/VizBin)

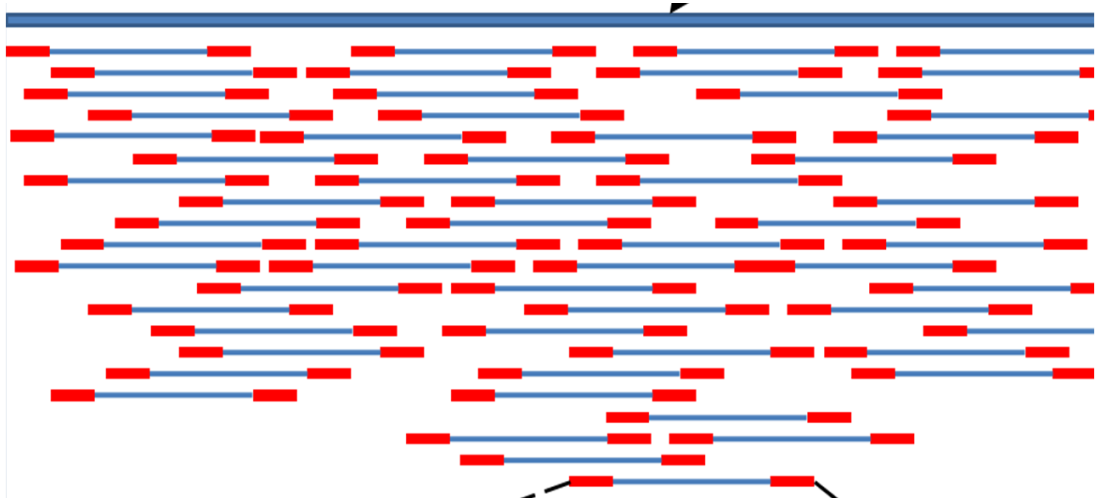


DTU_2017_1011718_1_MG_PT_PO_scaffolds.fasta

1 Quatify scaffolds with BBmap

- bbmap.sh in=DTU_2017_1011718_1_MG_PT_PO_R1.fastq.gz in2=DTU_2017_1011718_1_MG_PT_PO_R2.fastq.gz minid=0.90 threads=20 ref=scaffolds_gt1000bp.fa outm=mapped/mapped.sam overwrite=t nodisk=t

```
BBMap v36.x
Written by Brian Bushnell, from Dec. 2010 - present
Last modified July 6, 2016

Description:  Fast and accurate splice-aware read aligner.

To index:     bbmap.sh ref=<reference fasta>
To map:       bbmap.sh in=<reads> out=<output sam>
To map without writing an index:
    bbmap.sh ref=<reference fasta> in=<reads> out=<output sam> nodisk

nodisk=f                Set to true to build index in memory and write nothing
                        to disk except output.
ref=<file>              Specify the reference sequence.  Only do this ONCE,
                        when building the index (unless using 'nodisk').
in=<file>               Primary reads input; required parameter.
in2=<file>              For paired reads in two files.
minid=0.76              Approximate minimum alignment identity to look for.

threads=auto            (t) Set to number of threads desired.  By default, uses
                        all cores available.
outm=<file>             Write only mapped reads to this file.  Includes
                        unmapped paired reads with a mapped mate.
```



```
[pmun@g-12-l0002 group_test]$ samtools view mapped/mapped.sam | cut -f1-10 | head -n 10 | column -t
A00197:28:H5N3GDMXX:1:1101:29884:1485  1:N:0:GAGATTCC+GTACTGAC  83   23554  1532  42  1=1X142=1X5=                  =  1222  -460  TCCTGTATCTATAAATACTATTTTATTAGAATGCCTTACTGCCTCTGCCTCAAGCTTCTTCTGATGCATTGCTATTTTTATATAATCCCCGTACTCTAGGTTATCCTCATTTCCGCCGCATTCTTCATAAACATAATTTCTGCCGTATTC
A00197:28:H5N3GDMXX:1:1101:29884:1485  1:N:0:GAGATTCC+GTACTGAC  163  23554  1222  37  4=1D26=1X73=1X26=1X16=1X1=   =  1532  460   ATTCTTCTTCTATATATCATAAAACTCCCCTTTTATTATTTCCAAAGCCTTATCCAGCCTTTCTTCATAATTTCCCGATATCATTTTTACTTTATCGGCAATATAATATTTTTTCGAGTAATTCTGCCAGTAATTTTTTTCCTTTATTCAG
A00197:28:H5N3GDMXX:1:1101:5376:1501   1:N:0:GAGATTCC+GTACTGAC  83   24638  1051  44  48=1X101=                    =  690   -511  GTCCAAGTCGCCGTCGAAGGTGCCGACCTGCCCGCGCCCAACCAGCCAGAAATCATCGGCCACGGCCCGCAGCAAATGCCGGTCGTGGCTGACCAGCAGCACGGTGCCGTCAAAGTCGTTCAGCGCCATGGCCAGGGCCTCGCGCGTGGG
A00197:28:H5N3GDMXX:1:1101:5376:1501   1:N:0:GAGATTCC+GTACTGAC  163  24638  690   38  1=1X19=1X3=1X10=1X63=1X49=   =  1051  511   CGATGGCTTCGCTCAATTCCAACCAGCGTTCCTCACTGGCATCGAGTTCATCGCCCACGGCTTTGAGGCGTTTGCCCGCTTCGGCAATGTCTGCCGGGGGAAGGGGGGTGGCCAGTTTTTCTTCCAGGGCGGCTTTTTCTGCCCCCAGTT
A00197:28:H5N3GDMXX:1:1101:5520:1501   1:N:0:GAGATTCC+GTACTGAC  99   2649   868   45  150=                        =  868   150   GCCGTTAGTGATTGTAAGCTCCATGTAAAGCTTAGAATCTGCTCCACCATTCAAGTGTGCAATCTCTAACTCAGGATTCAAAATCTCGATATCTGAATCAACCTGAATATCGGCTGCTGTAACAACGCCTTCACCTTCGAATTCGATATA
A00197:28:H5N3GDMXX:1:1101:5520:1501   1:N:0:GAGATTCC+GTACTGAC  147  2649   868   45  150=                        =  868   -150  GCCGTTAGTGATTGTAAGCTCCATGTAAAGCTTAGAATCTGCTCCACCATTCAAGTGTGCAATCTCTAACTCAGGATTCAAAATCTCGATATCTGAATCAACCTGAATATCGGCTGCTGTAACAACGCCTTCACCTTCGAATTCGATATA
A00197:28:H5N3GDMXX:1:1101:9878:1501   1:N:0:GAGATTCC+GTACTGAC  83   1467   3663  45  150=                        =  3313  -500  AGGGTGAGCTGGTGGCGCGGTGGATTCTCGACAAAGCACATAACCAGAAGCCGTGTAAGCGTCTCTGGATTTCCTCTGTGACGGACAAGGCGATCCGTGAGGGTTTTGCCAATCTGAAGGATGCGAAAGAGTACGACAATCTCTACCGCG
A00197:28:H5N3GDMXX:1:1101:9878:1501   1:N:0:GAGATTCC+GTACTGAC  163  1467   3313  45  150=                        =  3663  500   AAAGAGCTATAAAAGGAGCAGACATTTTATGAAAACTCTGGTAATTGCAGAAAAACCATCCGTGGCACGCGACATTGCGCGCGTACTGGGGTGCAGCAAAAAAGAAAACGGAGCGCTGGAGGGAAACAAGTATATCGTTACCTGGGCGCT
A00197:28:H5N3GDMXX:1:1101:10474:1501  1:N:0:GAGATTCC+GTACTGAC  83   65299  901   39  37=1X15=1X7=2X6=1X80=        =  681   -370  CACCTCGAACTTCGGCTCGCCGGTCTTGCCGCTGGCCCTCAGGCCCACGCGCGGCTGCGCGGTGGCAGGCGAAGCGGCGGCGCTGCCCACGAGGTCGAGCTTGGCGATGTCGCCGCCCTTGTAGAACAGCGCGTAACGCTCGTCGACCTG
A00197:28:H5N3GDMXX:1:1101:10474:1501  1:N:0:GAGATTCC+GTACTGAC  163  65299  681   39  10=1X20=1X10=2X14=1X91=      =  901   370   TGTCGGCCAGGACGCCGTACTGCTCGGCCGTTGGCGGCATCAATTGCATCAAGCCGACGGCGCCCTTGGGCGACACGGCCTGCGGGTCAAAGCCCGATTCGGCGGCGATCACGGCTTTCAGCAACTCGTAATCGACCTTGTGCGTCTTGG
```

2 Make a sorted and indexed BAM file

- samtools view -bSh1 mapped/mapped.sam | samtools sort -m 20G -@ 3 > mapped/mapped.sort.bam

```
Usage: samtools view [options] <in.bam>|<in.sam>|<in.cram> [region ...]

Options:
  -b       output BAM
  -C       output CRAM (requires -T)
  -1       use fast BAM compression (implies -b)
  -u       uncompressed BAM output (implies -b)
  -h       include header in SAM output
  -H       print SAM header only (no alignments)
  -c       print only the count of matching records
  -o FILE  output file name [stdout]
  -U FILE  output reads not selected by filters to FILE [null]
  -t FILE  FILE listing reference names and lengths (see long help) [null]
  -L FILE  only include reads overlapping this BED FILE [null]
  -r STR   only include reads in read group STR [null]
  -R FILE  only include reads with read group listed in FILE [null]
  -q INT   only include reads with mapping quality >= INT [0]
  -l STR   only include reads in library STR [null]
  -m INT   only include reads with number of CIGAR operations consuming
           query sequence >= INT [0]
  -f INT   only include reads with all  of the FLAGs in INT present [0]
  -F INT   only include reads with none of the FLAGS in INT present [0]
  -G INT   only EXCLUDE reads with all  of the FLAGs in INT present [0]
  -s FLOAT subsample reads (given INT.FRAC option value, 0.FRAC is the
           fraction of templates/read pairs to keep; INT part sets seed)
  -M       use the multi-region iterator (increases the speed, removes
           duplicates and outputs the reads as they are ordered in the file)
  -x STR   read tag to strip (repeatable) [null]
  -B       collapse the backward CIGAR operation
  -?       print long help, including note about region specification
  -S       ignored (input format is auto-detected)
      --input-fmt-option OPT[=VAL]
               Specify a single input file format option in the form
               of OPTION or OPTION=VALUE
  -O, --output-fmt FORMAT[,OPT[=VAL]]...
               Specify output format (SAM, BAM, CRAM)
      --output-fmt-option OPT[=VAL]
               Specify a single output file format option in the form
               of OPTION or OPTION=VALUE
  -T, --reference FILE
               Reference sequence FASTA FILE [null]
  -@, --threads INT
               Number of additional threads to use [0]
```

```
Usage: samtools sort [options...] [in.bam]
Options:
  -l INT     Set compression level, from 0 (uncompressed) to 9 (best)
  -m INT     Set maximum memory per thread; suffix K/M/G recognized [768M]
  -n         Sort by read name
  -t TAG     Sort by value of TAG. Uses position as secondary index (or read name if -n is set)
  -o FILE    Write final output to FILE rather than standard output
  -T PREFIX  Write temporary files to PREFIX.nnnn.bam
      --input-fmt-option OPT[=VAL]
               Specify a single input file format option in the form
               of OPTION or OPTION=VALUE
  -O, --output-fmt FORMAT[,OPT[=VAL]]...
               Specify output format (SAM, BAM, CRAM)
      --output-fmt-option OPT[=VAL]
               Specify a single output file format option in the form
               of OPTION or OPTION=VALUE
      --reference FILE
               Reference sequence FASTA FILE [null]
  -@, --threads INT
               Number_of additional threads to use [0]
```

```
[pmun@g-12-l0002 group_test]$ samtools view mapped/mapped.sort.bam | cut -f1-10 | head -n 10
A00197:28:H5N3GDMXX:1:1108:17779:2440 1:N:0:GAGATTCC+GTACTGAC   165     0       1       0       *       =       1       0       ATACCATCATTCGGAACGGTTGTCAATGGTTATTTTGTTTATTTTGTAATTTATTCGGAATGGATTCGGGCGGGTGGATGGGCTGTTACAATTTGTTACTCATTATTTGTTTACAAAGGTATGAATATATCGTATAATTGTGGTTCAAAT
A00197:28:H5N3GDMXX:1:1129:21106:30405 1:N:0:GAGATTCC+GTACTGAC   99      0       1       37      15S135= =       229     378     GGATGGGCTGTTACAATTTGTTACTCATTATTTGTTTACAAAGGTATGAATATATCGTATAATTGTGGTTCAAATGGGAAATGAGGTTTATAATGGAACATATATATCAAGAATATACAACATCGGATGGTATATCAGAAGAAAAGAAGC
A00197:28:H5N3GDMXX:1:1132:23050:25473 1:N:0:GAGATTCC+GTACTGAC   99      0       1       43      4S146=  =       385     534     TACAATTTGTTACTCATTATTTGTTTACAAAGGTATGAATATATCGTATAATTGTGGTTCAAATGGGAAATGAGGTTTATAATGGAACATATATATCAAGAATATACAACATCGGATGGTATATCAGAAGAAAAGAAGCATAATGCATTA
A00197:28:H5N3GDMXX:1:1142:16703:9189 1:N:0:GAGATTCC+GTACTGAC   165     0       1       0       *       =       1       0       TATACCATCATTCGGAACGGTTGTCAATGGTTATTTTGTTTATTTTGTAATTTATTCGGAATGGATTCGGGCGGGTGGATGGGCTGTTACAATTTGTTACTCATTATTTGTTTACAAAGGTATGAATATATCGTATAATTGTGGTTCAAA
A00197:28:H5N3GDMXX:1:1148:20039:33004 1:N:0:GAGATTCC+GTACTGAC   165     0       1       0       *       =       1       0       CGGTTGTCAATGGTTATTTTGTTTATTTTGTAATTTATTCGGAATGGATTCGGGCGGGTGGATGGGCTGTTACAATTTGTTACTCATTATTTGTTTACAAAGGTATGAATATATCGTATAATTGTGGTTCAAATGGGAAATGAGGTTTAT
A00197:28:H5N3GDMXX:1:1154:13783:11115 1:N:0:GAGATTCC+GTACTGAC   163     0       1       35      20S130= =       384     533     CGGGTGGATGGGCTGTTACAATTTGTTACTCATTATTTGTTTACAAAGGTATGAATATATCGTATAATTGTGGTTCAAATGGGAAATGAGGTTTATAATGGAACATATATATCAAGAATATACAACATCGGATGGTATATCAGAAGAAAA
A00197:28:H5N3GDMXX:1:1158:12463:17315 1:N:0:GAGATTCC+GTACTGAC   99      0       1       31      27S123= =       175     324     ATTCGGGCGGGTGGATGGGCTGTTACAATTTGTTACTCATTATTTGTTTACAAAGGTATGAATATATCGTATAATTGTGGTTCAAATGGGAAATGAGGTTTATAATGGAACATATATATCAAGAATATACAACATCGGATGGTATATCAG
A00197:28:H5N3GDMXX:1:1187:6867:11287 1:N:0:GAGATTCC+GTACTGAC   101     0       1       0       *       =       1       0       ATGCATTATACCATCATTCGGAACGGTTGTCAATGGTTATTTTGTTTATTTTGTAATTTATTCGGAATGGATTCGGGCGGGTGGATGGGCTGTTACAATTTGTTACTCATTATTTGTTTACAAAGGTATGAATATATCGTATAATTGTGG
A00197:28:H5N3GDMXX:1:1187:9164:13980 1:N:0:GAGATTCC+GTACTGAC   101     0       1       0       *       =       1       0       ATGCATTATACCATCATTCGGAACGGTTGTCAATGGTTATTTTGTTTATTTTGTAATTTATTCGGAATGGATTCGGGCGGGTGGATGGGCTGTTACAATTTGTTACTCATTATTTGTTTACAAAGGTATGAATATATCGTATAATTGTGG
A00197:28:H5N3GDMXX:1:1269:26422:8015 1:N:0:GAGATTCC+GTACTGAC   99      0       1       42      6S144=  =       91      240     GTTACAATTTGTTACTCATTATTTGTTTACAAAGGTATGAATATATCGTATAATTGTGGTTCAAATGGGAAATGAGGTTTATAATGGAACATATATATCAAGAATATACAACATCGGATGGTATATCAGAAGAAAAGAAGCATAATGCAT
```

3 Make a coverage table

- jgi_summarize_bam_contig_depths mapped/mapped.sort.bam --outputDepth mapped/coverage.txt

```
[pmun@g-12-l0002 group_test]$ jgi_summarize_bam_contig_depths -h
Usage: jgi_summarize_bam_contig_depths <options> sortedBam1 [ sortedBam2 ...]
where options include:
        --outputDepth       arg  The file to put the contig by bam depth matrix (default: STDOUT)
        --percentIdentity   arg  The minimum end-to-end % identity of qualifying reads (default: 97)
        --pairedContigs     arg  The file to output the sparse matrix of contigs which paired reads span (default: none)
        --unmappedFastq     arg  The prefix to output unmapped reads from each bam file suffixed by 'bamfile.bam.fastq.gz'
        --noIntraDepthVariance   Do not include variance from mean depth along the contig
        --showDepth              Output a .depth file per bam for each contig base
        --minMapQual        arg  The minimum mapping quality necessary to count the read as mapped (default: 0)
        --weightMapQual     arg  Weight per-base depth based on the MQ of the read (i.e uniqueness) (default: 0.0 (disabled))
        --includeEdgeBases       When calculating depth & variance, include the 1-readlength edges (off by default)
        --maxEdgeBases           When calculating depth & variance, and not --includeEdgeBases, the maximum length (default:75)
        --referenceFasta    arg  The reference file.  (It must be the same fasta that bams used)

Options that require a --referenceFasta
        --outputGC          arg  The file to print the gc coverage histogram
        --gcWindow          arg  The sliding window size for GC calculations
        --outputReadStats   arg  The file to print the per read statistics
        --outputKmers       arg  The file to print the perfect kmer counts

Options to control shredding contigs that are under represented by the reads
        --shredLength       arg  The maximum length of the shreds
        --shredDepth        arg  The depth to generate overlapping shreds
        --minContigLength   arg  The mimimum length of contig to include for mapping and shredding
        --minContigDepth    arg  The minimum depth along contig at which to break the contig
```

3 Coverage Table

contigName	contigLen	totalAvgDepth	mapped.sort.bam	mapped.sort.bam-var
0	117507	578.735	578.735	27475.2
1	100978	19.6234	19.6234	33.6758
2	99770	11.7375	11.7375	20.1838
3	93846	45.4139	45.4139	108.155
4	93791	13.4738	13.4738	22.1335
5	93614	11.6939	11.6939	18.9678
6	87329	12.5853	12.5853	22.0302
7	84431	13.1622	13.1622	22.7428
8	78310	14.4304	14.4304	25.5052
9	75851	15.0533	15.0533	29.356
10	75525	13.4923	13.4923	22.1386
11	72210	13.9444	13.9444	25.1159
12	63361	17.667	17.667	28.966
13	61513	11.8235	11.8235	17.309
14	60700	11.5295	11.5295	17.9902
15	60376	22.0164	22.0164	33.6955
16	60329	13.0686	13.0686	21.1894
17	59992	19.5848	19.5848	32.7748
18	57935	17.1185	17.1185	40.9718

3 Coverage Table – with multiple libraries

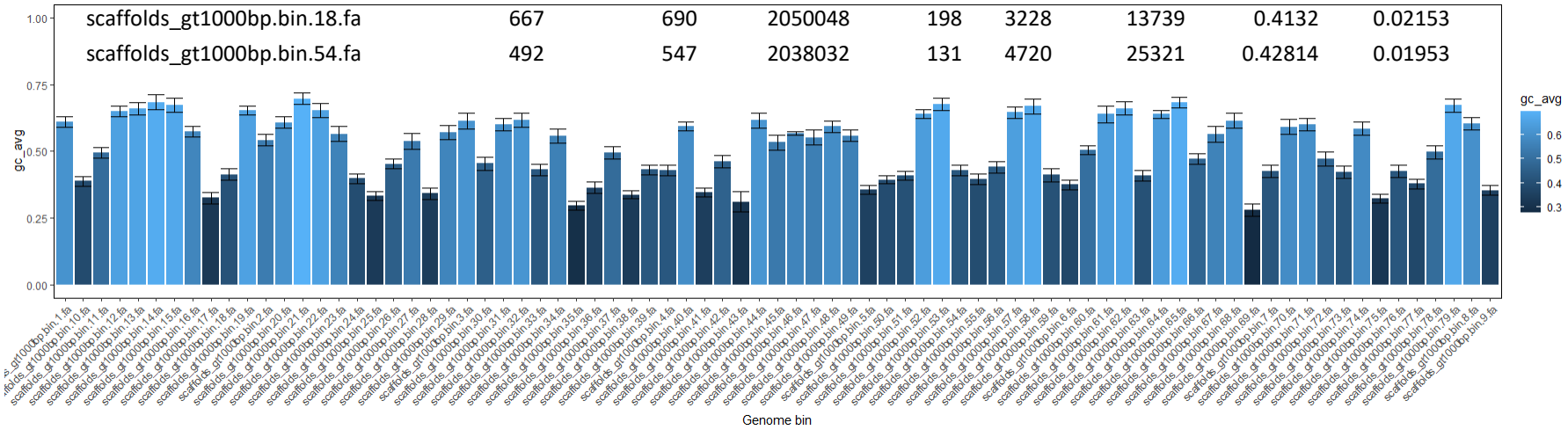
contigName	contigLen	totalAvgDepth	sample1.sort.bam	sample1.sort.bam-var	sample2.sort.bam	sample2.sort.bam-var	sample3.sort.bam	sample3.sort.bam-var
0	117507	578.735	578.735	27475.2	578.735	27475.2	578.735	27475.2
1	100978	19.6234	19.6234	33.6758	19.6234	33.6758	19.6234	33.6758
2	99770	11.7375	11.7375	20.1838	11.7375	20.1838	11.7375	20.1838
3	93846	45.4139	45.4139	108.155	45.4139	108.155	45.4139	108.155
4	93791	13.4738	13.4738	22.1335	13.4738	22.1335	13.4738	22.1335
5	93614	11.6939	11.6939	18.9678	11.6939	18.9678	11.6939	18.9678
6	87329	12.5853	12.5853	22.0302	12.5853	22.0302	12.5853	22.0302
7	84431	13.1622	13.1622	22.7428	13.1622	22.7428	13.1622	22.7428
8	78310	14.4304	14.4304	25.5052	14.4304	25.5052	14.4304	25.5052
9	75851	15.0533	15.0533	29.356	15.0533	29.356	15.0533	29.356
10	75525	13.4923	13.4923	22.1386	13.4923	22.1386	13.4923	22.1386
11	72210	13.9444	13.9444	25.1159	13.9444	25.1159	13.9444	25.1159
12	63361	17.667	17.667	28.966	17.667	28.966	17.667	28.966
13	61513	11.8235	11.8235	17.309	11.8235	17.309	11.8235	17.309
14	60700	11.5295	11.5295	17.9902	11.5295	17.9902	11.5295	17.9902
15	60376	22.0164	22.0164	33.6955	22.0164	33.6955	22.0164	33.6955
16	60329	13.0686	13.0686	21.1894	13.0686	21.1894	13.0686	21.1894
17	59992	19.5848	19.5848	32.7748	19.5848	32.7748	19.5848	32.7748
18	57935	17.1185	17.1185	40.9718	17.1185	40.9718	17.1185	40.9718

4 MetaBat binning

```
[pmun@g-12-l0002 group_test]$ metabat -h

MetaBAT: Metagenome Binning based on Abundance and Tetranucleotide frequency (version 2.12.1; Aug 31 2017 21:02:54)
by Don Kang (ddkang@lbl.gov), Feng Li, Jeff Froula, Rob Egan, and Zhong Wang (zhongwang@lbl.gov)

Allowed options:
  -h [ --help ]                       produce help message
  -i [ --inFile ] arg                 Contigs in (gzipped) fasta file format [Mandatory]
  -o [ --outFile ] arg                Base file name and path for each bin. The default output is fasta format.
                                      Use -l option to output only contig names [Mandatory].
  -a [ --abdFile ] arg                A file having mean and variance of base coverage depth (tab delimited;
                                      the first column should be contig names, and the first row will be
                                      considered as the header and be skipped) [Optional].
  -m [ --minContig ] arg (=2500)      Minimum size of a contig for binning (should be >=1500).
  --maxP arg (=95)                    Percentage of 'good' contigs considered for binning decided by connection
                                      among contigs. The greater, the more sensitive.
  --minS arg (=60)                    Minimum score of a edge for binning (should be between 1 and 99). The
                                      greater, the more specific.
  --maxEdges arg (=200)               Maximum number of edges per node. The greater, the more sensitive.
  --pTNF arg (=0)                     TNF probability cutoff for building TNF graph. Use it to skip the
                                      preparation step. (0: auto).
  --noAdd                             Turning off additional binning for lost or small contigs.
  --cvExt                             When a coverage file without variance (from third party tools) is used
                                      instead of abdFile from jgi_summarize_bam_contig_depths.
  -x [ --minCV ] arg (=1)             Minimum mean coverage of a contig in each library for binning.
  --minCVSum arg (=1)                 Minimum total effective mean coverage of a contig (sum of depth over
                                      minCV) for binning.
  -s [ --minClsSize ] arg (=200000)   Minimum size of a bin as the output.
  -t [ --numThreads ] arg (=0)        Number of threads to use (0: use all cores).
  -l [ --onlyLabel ]                  Output only sequence labels as a list in a column without sequences.
  --saveCls                           Save cluster memberships as a matrix format
  --unbinned                          Generate [outFile].unbinned.fa file for unbinned contigs
  --noBinOut                          No bin output. Usually combined with --saveCls to check only contig
                                      memberships
  --seed arg (=0)                     For exact reproducibility. (0: use random seed)
  -d [ --debug ]                      Debug output
  -v [ --verbose ]                    Verbose output
```

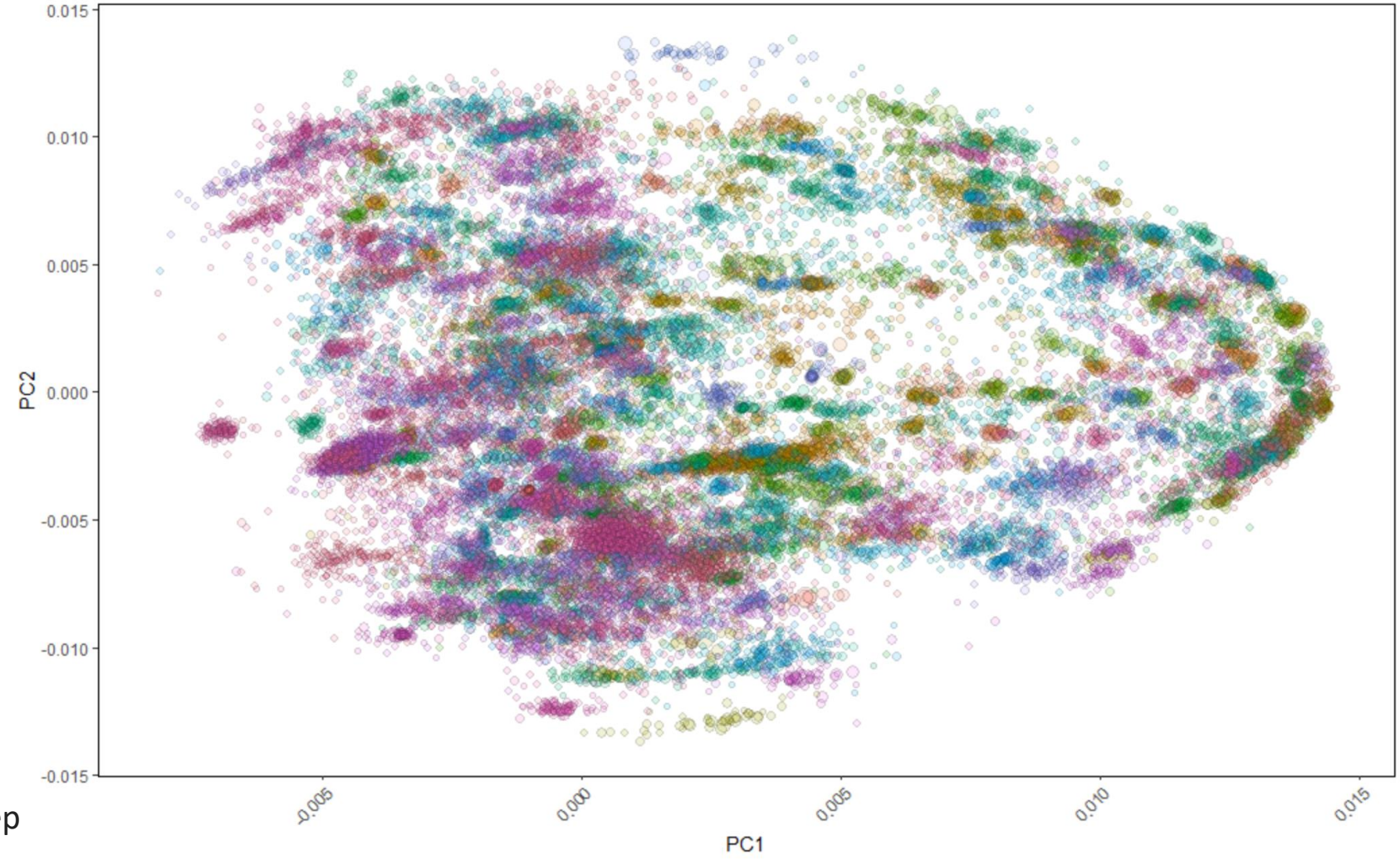
4 MetaBat binning

filename	n_scaffolds	n_contigs	scaf_bp	N50	L50	scaf_max	gc_avg	gc_std
scaffolds_gt1000bp.bin.43.fa	5921	6192	17039999	1623	2870	56969	0.31029	0.03766
scaffolds_gt1000bp.bin.31.fa	2001	2068	4326132	764	2114	9901	0.60113	0.02298
scaffolds_gt1000bp.bin.73.fa	1525	1592	4281230	475	2845	35604	0.42156	0.02338
scaffolds_gt1000bp.bin.27.fa	1600	1652	4215778	553	2709	9982	0.53841	0.02893
scaffolds_gt1000bp.bin.70.fa	1064	1105	3873002	272	4131	24545	0.59214	0.0276
scaffolds_gt1000bp.bin.16.fa	208	214	3623157	33	37072	99770	0.5726	0.01991
scaffolds_gt1000bp.bin.15.fa	1380	1444	3332777	463	2320	37527	0.67209	0.02612
scaffolds_gt1000bp.bin.39.fa	919	973	2809677	269	3146	34190	0.43052	0.01933
scaffolds_gt1000bp.bin.61.fa	948	971	2767226	283	3090	56756	0.63941	0.03077
scaffolds_gt1000bp.bin.21.fa	848	884	2581273	262	3217	15895	0.69758	0.02097
scaffolds_gt1000bp.bin.28.fa	885	910	2555610	291	3065	10107	0.34206	0.02138
scaffolds_gt1000bp.bin.22.fa	1036	1086	2478729	360	2409	10658	0.6525	0.02648
scaffolds_gt1000bp.bin.47.fa	725	744	2172057	224	3146	23290	0.55169	0.02775
scaffolds_gt1000bp.bin.18.fa	667	690	2050048	198	3228	13739	0.4132	0.02153
scaffolds_gt1000bp.bin.54.fa	492	547	2038032	131	4720	25321	0.42814	0.01953




```
-rw-rw-r-- 1 pmun co_23260 1.3M Oct 21 13:41 scaffolds_gt1000bp.bin
-rw-rw-r-- 1 pmun co_23260 628K Oct 21 13:41 scaffolds_gt1000bp.bin.10.fa
-rw-rw-r-- 1 pmun co_23260 1.6M Oct 21 13:41 scaffolds_gt1000bp.bin.11.fa
-rw-rw-r-- 1 pmun co_23260 565K Oct 21 13:41 scaffolds_gt1000bp.bin.12.fa
-rw-rw-r-- 1 pmun co_23260 385K Oct 21 13:41 scaffolds_gt1000bp.bin.13.fa
-rw-rw-r-- 1 pmun co_23260 1.6M Oct 21 13:41 scaffolds_gt1000bp.bin.14.fa
-rw-rw-r-- 1 pmun co_23260 3.3M Oct 21 13:41 scaffolds_gt1000bp.bin.15.fa
-rw-rw-r-- 1 pmun co_23260 3.6M Oct 21 13:41 scaffolds_gt1000bp.bin.16.fa
-rw-rw-r-- 1 pmun co_23260 1.1M Oct 21 13:41 scaffolds_gt1000bp.bin.17.fa
-rw-rw-r-- 1 pmun co_23260 2.0M Oct 21 13:41 scaffolds_gt1000bp.bin.18.fa
-rw-rw-r-- 1 pmun co_23260 256K Oct 21 13:41 scaffolds_gt1000bp.bin.19.fa
-rw-rw-r-- 1 pmun co_23260 431K Oct 21 13:41 scaffolds_gt1000bp.bin.1.fa
-rw-rw-r-- 1 pmun co_23260 972K Oct 21 13:41 scaffolds_gt1000bp.bin.20.fa
-rw-rw-r-- 1 pmun co_23260 2.6M Oct 21 13:41 scaffolds_gt1000bp.bin.21.fa
-rw-rw-r-- 1 pmun co_23260 2.5M Oct 21 13:41 scaffolds_gt1000bp.bin.22.fa
-rw-rw-r-- 1 pmun co_23260 1.7M Oct 21 13:41 scaffolds_gt1000bp.bin.23.fa
-rw-rw-r-- 1 pmun co_23260 702K Oct 21 13:41 scaffolds_gt1000bp.bin.24.fa
-rw-rw-r-- 1 pmun co_23260 407K Oct 21 13:41 scaffolds_gt1000bp.bin.25.fa
-rw-rw-r-- 1 pmun co_23260 333K Oct 21 13:41 scaffolds_gt1000bp.bin.26.fa
-rw-rw-r-- 1 pmun co_23260 4.1M Oct 21 13:41 scaffolds_gt1000bp.bin.27.fa
-rw-rw-r-- 1 pmun co_23260 2.5M Oct 21 13:41 scaffolds_gt1000bp.bin.28.fa
-rw-rw-r-- 1 pmun co_23260 1.9M Oct 21 13:41 scaffolds_gt1000bp.bin.29.fa
-rw-rw-r-- 1 pmun co_23260 866K Oct 21 13:41 scaffolds_gt1000bp.bin.2.fa
-rw-rw-r-- 1 pmun co_23260 1.9M Oct 21 13:41 scaffolds_gt1000bp.bin.30.fa
-rw-rw-r-- 1 pmun co_23260 4.3M Oct 21 13:41 scaffolds_gt1000bp.bin.31.fa
-rw-rw-r-- 1 pmun co_23260 1.1M Oct 21 13:41 scaffolds_gt1000bp.bin.32.fa
-rw-rw-r-- 1 pmun co_23260 1.3M Oct 21 13:41 scaffolds_gt1000bp.bin.33.fa
-rw-rw-r-- 1 pmun co_23260 1.8M Oct 21 13:41 scaffolds_gt1000bp.bin.34.fa
-rw-rw-r-- 1 pmun co_23260 1.5M Oct 21 13:41 scaffolds_gt1000bp.bin.35.fa
-rw-rw-r-- 1 pmun co_23260 407K Oct 21 13:41 scaffolds_gt1000bp.bin.36.fa
-rw-rw-r-- 1 pmun co_23260 1.9M Oct 21 13:41 scaffolds_gt1000bp.bin.37.fa
-rw-rw-r-- 1 pmun co_23260 280K Oct 21 13:41 scaffolds_gt1000bp.bin.38.fa
-rw-rw-r-- 1 pmun co_23260 2.8M Oct 21 13:41 scaffolds_gt1000bp.bin.39.fa
-rw-rw-r-- 1 pmun co_23260 862K Oct 21 13:41 scaffolds_gt1000bp.bin.3.fa
-rw-rw-r-- 1 pmun co_23260 515K Oct 21 13:41 scaffolds_gt1000bp.bin.40.fa
-rw-rw-r-- 1 pmun co_23260 679K Oct 21 13:41 scaffolds_gt1000bp.bin.41.fa
-rw-rw-r-- 1 pmun co_23260 647K Oct 21 13:41 scaffolds_gt1000bp.bin.42.fa
-rw-rw-r-- 1 pmun co_23260  17M Oct 21 13:41 scaffolds_gt1000bp.bin.43.fa
-rw-rw-r-- 1 pmun co_23260 1.1M Oct 21 13:41 scaffolds_gt1000bp.bin.44.fa
-rw-rw-r-- 1 pmun co_23260 1.3M Oct 21 13:41 scaffolds_gt1000bp.bin.45.fa
-rw-rw-r-- 1 pmun co_23260 264K Oct 21 13:41 scaffolds_gt1000bp.bin.46.fa
-rw-rw-r-- 1 pmun co_23260 2.2M Oct 21 13:41 scaffolds_gt1000bp.bin.47.fa
-rw-rw-r-- 1 pmun co_23260 434K Oct 21 13:41 scaffolds_gt1000bp.bin.48.fa
-rw-rw-r-- 1 pmun co_23260 1.4M Oct 21 13:41 scaffolds_gt1000bp.bin.49.fa
-rw-rw-r-- 1 pmun co_23260 883K Oct 21 13:41 scaffolds_gt1000bp.bin.4.fa
-rw-rw-r-- 1 pmun co_23260 267K Oct 21 13:41 scaffolds_gt1000bp.bin.50.fa
-rw-rw-r-- 1 pmun co_23260 312K Oct 21 13:41 scaffolds_gt1000bp.bin.51.fa
-rw-rw-r-- 1 pmun co_23260 734K Oct 21 13:41 scaffolds_gt1000bp.bin.52.fa
-rw-rw-r-- 1 pmun co_23260 203K Oct 21 13:41 scaffolds_gt1000bp.bin.53.fa
-rw-rw-r-- 1 pmun co_23260 2.0M Oct 21 13:41 scaffolds_gt1000bp.bin.54.fa
-rw-rw-r-- 1 pmun co_23260 927K Oct 21 13:41 scaffolds_gt1000bp.bin.55.fa
-rw-rw-r-- 1 pmun co_23260 1.8M Oct 21 13:41 scaffolds_gt1000bp.bin.56.fa
-rw-rw-r-- 1 pmun co_23260 226K Oct 21 13:41 scaffolds_gt1000bp.bin.57.fa
-rw-rw-r-- 1 pmun co_23260 656K Oct 21 13:41 scaffolds_gt1000bp.bin.58.fa
-rw-rw-r-- 1 pmun co_23260 330K Oct 21 13:41 scaffolds_gt1000bp.bin.59.fa
-rw-rw-r-- 1 pmun co_23260 309K Oct 21 13:41 scaffolds_gt1000bp.bin.5.fa
-rw-rw-r-- 1 pmun co_23260 202K Oct 21 13:41 scaffolds_gt1000bp.bin.60.fa
-rw-rw-r-- 1 pmun co_23260 2.7M Oct 21 13:41 scaffolds_gt1000bp.bin.61.fa
-rw-rw-r-- 1 pmun co_23260 1.7M Oct 21 13:41 scaffolds_gt1000bp.bin.62.fa
-rw-rw-r-- 1 pmun co_23260 1.7M Oct 21 13:41 scaffolds_gt1000bp.bin.63.fa
-rw-rw-r-- 1 pmun co_23260 273K Oct 21 13:41 scaffolds_gt1000bp.bin.64.fa
-rw-rw-r-- 1 pmun co_23260 294K Oct 21 13:41 scaffolds_gt1000bp.bin.65.fa
-rw-rw-r-- 1 pmun co_23260 513K Oct 21 13:41 scaffolds_gt1000bp.bin.66.fa
-rw-rw-r-- 1 pmun co_23260 1.1M Oct 21 13:41 scaffolds_gt1000bp.bin.67.fa
-rw-rw-r-- 1 pmun co_23260 491K Oct 21 13:41 scaffolds_gt1000bp.bin.68.fa
-rw-rw-r-- 1 pmun co_23260 732K Oct 21 13:41 scaffolds_gt1000bp.bin.69.fa
-rw-rw-r-- 1 pmun co_23260 500K Oct 21 13:41 scaffolds_gt1000bp.bin.6.fa
-rw-rw-r-- 1 pmun co_23260 3.8M Oct 21 13:41 scaffolds_gt1000bp.bin.70.fa
-rw-rw-r-- 1 pmun co_23260 1.6M Oct 21 13:41 scaffolds_gt1000bp.bin.71.fa
-rw-rw-r-- 1 pmun co_23260 1.9M Oct 21 13:41 scaffolds_gt1000bp.bin.72.fa
-rw-rw-r-- 1 pmun co_23260 4.2M Oct 21 13:41 scaffolds_gt1000bp.bin.73.fa
-rw-rw-r-- 1 pmun co_23260 1.3M Oct 21 13:41 scaffolds_gt1000bp.bin.74.fa
-rw-rw-r-- 1 pmun co_23260 1.1M Oct 21 13:41 scaffolds_gt1000bp.bin.75.fa
-rw-rw-r-- 1 pmun co_23260 814K Oct 21 13:41 scaffolds_gt1000bp.bin.76.fa
-rw-rw-r-- 1 pmun co_23260 868K Oct 21 13:41 scaffolds_gt1000bp.bin.77.fa
-rw-rw-r-- 1 pmun co_23260 801K Oct 21 13:41 scaffolds_gt1000bp.bin.78.fa
-rw-rw-r-- 1 pmun co_23260 1.7M Oct 21 13:41 scaffolds_gt1000bp.bin.79.fa
-rw-rw-r-- 1 pmun co_23260 1.1M Oct 21 13:41 scaffolds_gt1000bp.bin.7.fa
-rw-rw-r-- 1 pmun co_23260 1.8M Oct 21 13:41 scaffolds_gt1000bp.bin.8.fa
-rw-rw-r-- 1 pmun co_23260 288K Oct 21 13:41 scaffolds_gt1000bp.bin.9.fa
```

MetaBat2 bins on pig feces co-assembly



Munk et al. Manuscript in prep

Utilizing multiple binners

- Different binners have different strengths
- Genomes missed by one binner, might be identified by another
- Approaches to merge results from multiple binners have been developed

Recovery of genomes from metagenomes via a dereplication, aggregation and scoring strategy

Christian M. K. Sieber, Alexander J. Probst, Allison Sharrar, Brian C. Thomas, Matthias Hess, Susannah G. Tringe & Jillian F. Banfield

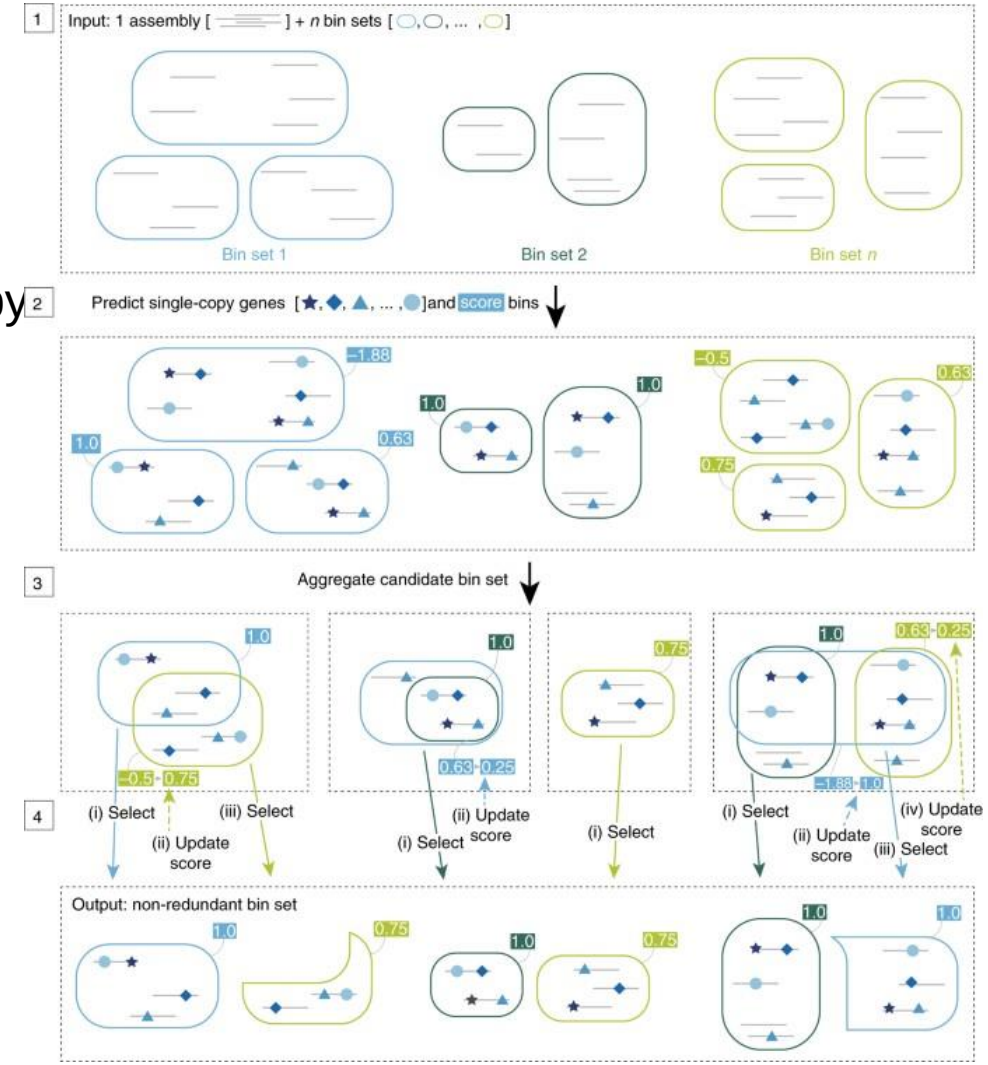
Nature Microbiology **3**, 836–843 (2018) | Cite this article

26k Accesses | **195** Citations | **192** Altmetric | Metrics

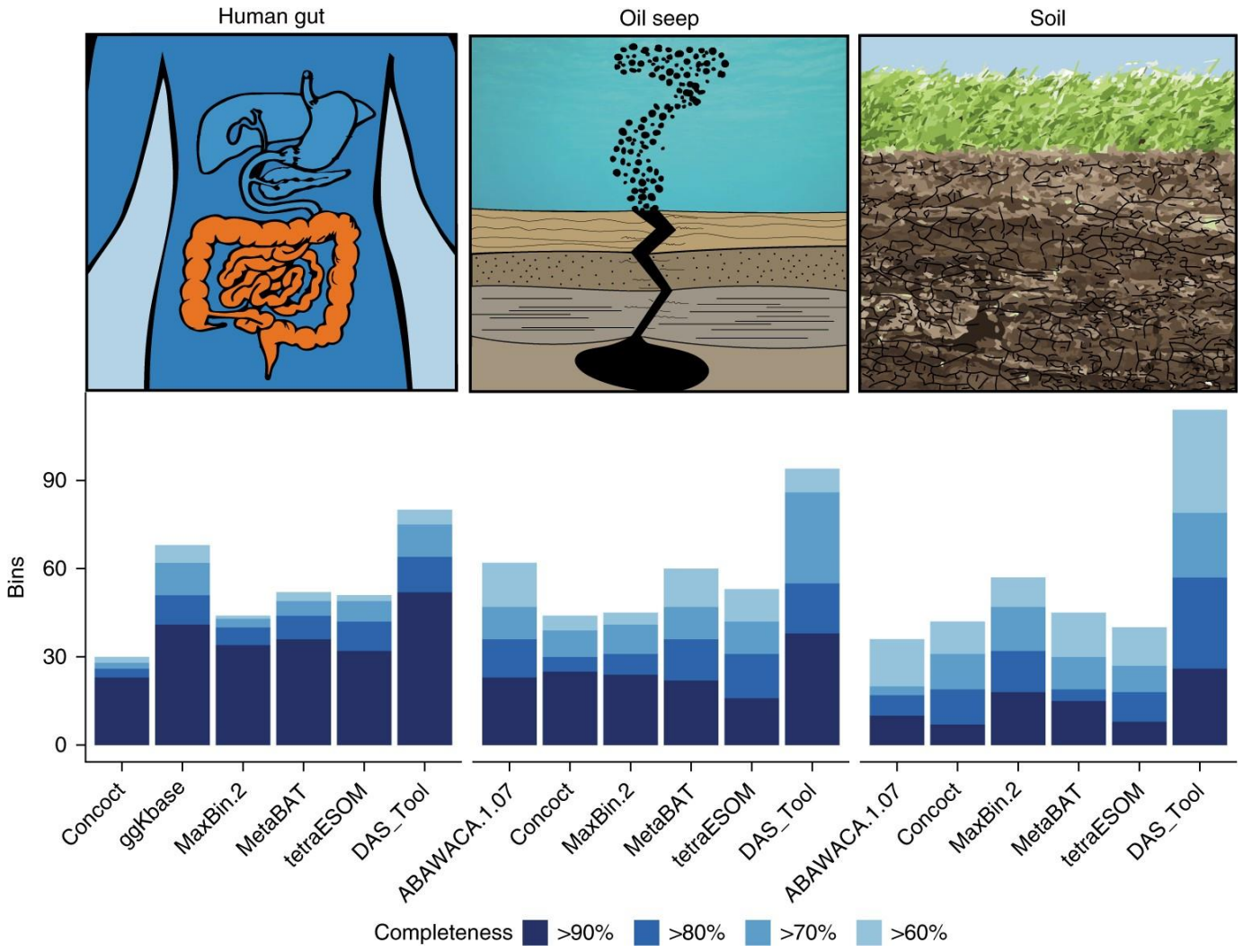
Abstract

Microbial communities are critical to ecosystem function. A key objective of metagenomic studies is to analyse organism-specific metabolic pathways and reconstruct community interaction networks. This requires accurate assignment of assembled genome fragments to genomes. Existing binning methods often fail to reconstruct a reasonable number of genomes and report many bins of low quality and completeness. Furthermore, the performance of existing algorithms varies between samples and biotopes. Here, we present a dereplication, aggregation and scoring strategy, DAS Tool, that combines the strengths of a flexible set of established binning algorithms. DAS Tool applied to a constructed community generated more accurate bins than any automated method. Indeed, when applied to environmental and host-associated samples of different complexity, DAS Tool recovered substantially more near-complete genomes, including previously unreported lineages, than any single binning method alone. The ability to reconstruct many near-complete genomes from metagenomics data will greatly advance genome-centric analyses of ecosystems.

Step 1: Scaffolds binned using different methods
Step 2: Single-copy genes are predicted on scaffolds and each bin is scored.
Step 3: Aggregation of redundant candidate bin set from all binning predictions.
Step 4: Iterative selection of high-scoring bins and updating of scores of remaining partial candidate bins.



DAS Tool


Human gut
Oil seep
Soil
Bins
90
60
30
0
Concoct
ggKbase
MaxBin.2
MetaBAT
tetraESOM
DAS_Tool
ABAWACA.1.07
Concoct
MaxBin.2
MetaBAT
tetraESOM
DAS_Tool
ABAWACA.1.07
Concoct
MaxBin.2
MetaBAT
tetraESOM
DAS_Tool
Completeness
>90%
>80%
>70%
>60%

VAMB

nature biotechnology
LETTERS
https://doi.org/10.1038/s41587-020-00777-4
Check for updates

Improved metagenome binning and assembly using deep variational autoencoders

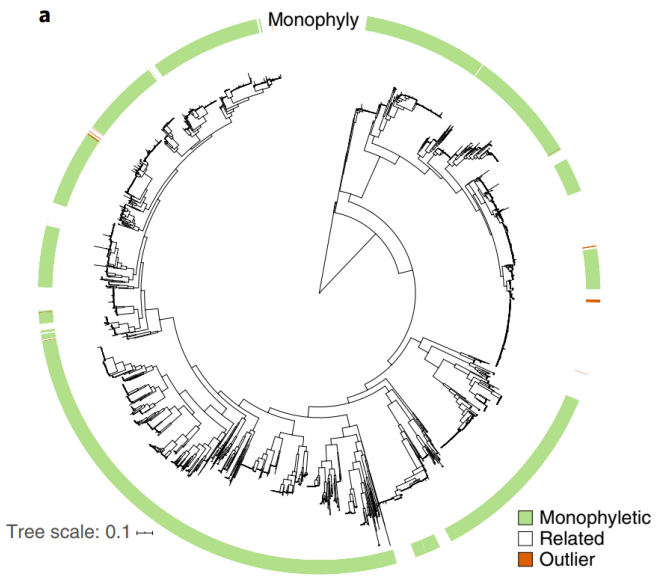
Jakob Nybo Nissen[1,2], Joachim Johansen[2], Rosa Lundbye Allesøe[2], Casper Kaae Sønderby[3], Jose Juan Almagro Armenteros[1], Christopher Heje Grønbech[3,4], Lars Juhl Jensen[2], Henrik Bjørn Nielsen[5], Thomas Nordahl Petersen[6], Ole Winther[3,4,7] and Simon Rasmussen[2]✉

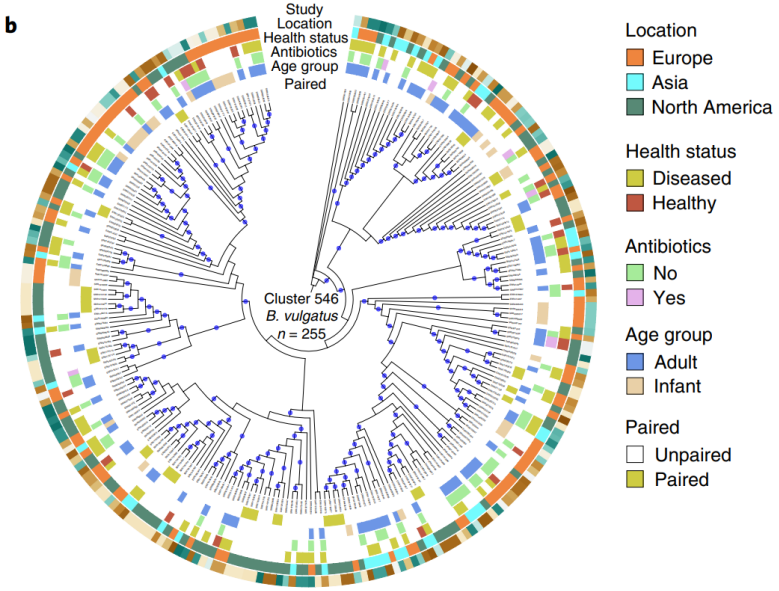
Despite recent advances in metagenomic binning, reconstruction of microbial species from metagenomics data remains challenging. Here we develop variational autoencoders for metagenomic binning (VAMB), a program that uses deep variational autoencoders to encode sequence coabundance and *k*-mer distribution information before clustering. We show that a variational autoencoder is able to integrate these two distinct data types without any previous knowledge of the datasets. VAMB outperforms existing state-of-the-art binners, reconstructing 29–98% and 45% more near-complete (NC) genomes on simulated and real data, respectively. Furthermore, VAMB is able to separate closely related strains up to 99.5% average nucleotide identity (ANI), and reconstructed 255 and 91 NC *Bacteroides vulgatus* and *Bacteroides dorei* sample-specific genomes as two distinct clusters from a dataset of 1,000 human gut microbiome samples. We use 2,606 NC bins from this dataset to show that species of the human gut microbiome have different geographical distribution patterns. VAMB can be run on standard hardware and is freely available at https://github.com/RasmussenLab/vamb.

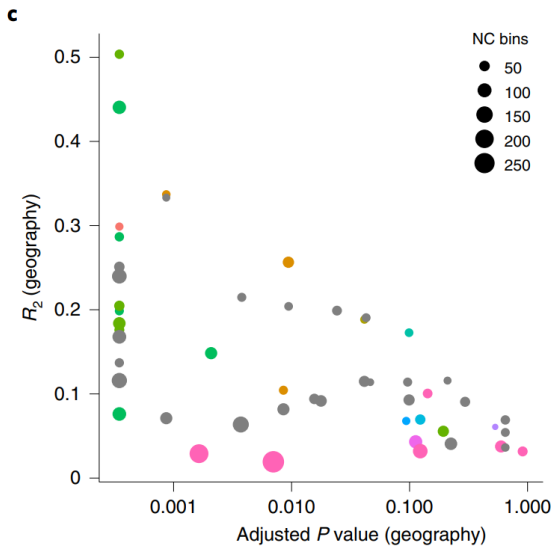
Metagenomic binning is the process of grouping metagenomic sequences by their organism of origin[1,2]. In metagenomic studies, binning allows the reconstruction of known and unknown genomes, enabling a broad description of the community and creating a starting point for further analysis of the organisms[3]. We developed a binning tool that uses deep learning in the form of variational autoencoders (VAE)[4,5] that integrates coabundance[6] and *k*-mer composition[7] data from metagenomics de novo assemblies and clusters the resulting latent representation into genome clusters and sample-specific bins. Our approach leverages multiple samples while simultaneously avoiding between-sample chimeras. It outperforms commonly used single-sample binning approaches by reconstructing 29–98% more NC genomes from simulated datasets, as well as 45% more NC genomes from a dataset of 1,000 human gut microbiome samples. Furthermore, our clustering method automatically groups per-sample bins into clusters with high taxonomic consistency, allowing precise strain-resolution taxonomic profiling.

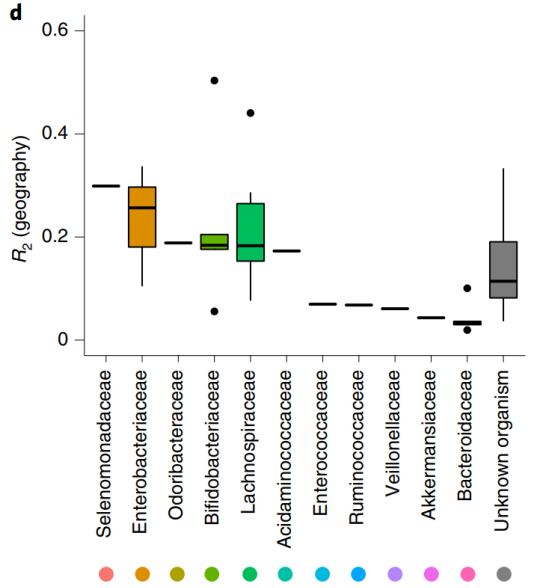
Earlier work on metagenomics binning has mainly relied on the principles that DNA sequences originating from the same organism will have high covariance of their abundance signal across samples (coabundance) and that they share similar patterns of *k*-mer usage in their DNA (for example, 2–5 mer)[7–15]. Several attempts have been made to reconstruct thousands of microbial species from massive metagenomics datasets[16–18], independently assembling and binning each sample into genomes. These simple workflows allow for parallel analysis of samples, but do not leverage coabundance. Typical workflows using coabundance deal with sequence redundancy by either coassembling distinct samples or deduplicating sequences before binning[6,8–12]. This leads to intersample chimeric genomes that do not exist, which is especially problematic when strain-level variation can have important biological implications[19]. Furthermore, none of the existing methods leverage deep learning.

The main difference between our method, VAMB, and others is that it utilizes an unsupervised deep learning approach known as a VAE[10,18]. Second, our approach clusters the combined contig dataset from all samples without any preclustering or homology reduction and applies a strategy for splitting genome clusters after clustering (Supplementary Figs. 1–4). When applying this approach, which we term 'multisplit', each cluster should correspond to an organism and each bin in a cluster to a per-sample representation of the genome of that organism. To demonstrate the performance of VAMB compared to other binners, we benchmarked VAMB, Canopy[6], MetaBAT2 (ref. [12]) and MaxBin2 (ref. [11]) on five synthetic datasets from Critical Assessment of Metagenomic Interpretation (CAMI)[20] and one semisynthetic dataset from MetaHIT[21] samples (Supplementary Table 1). We assessed binning performance by counting the number of NC (>90% recall and >95% precision) genomes reconstructed as done in previous work[19,22]. VAMB reconstructed 29–98% more NC genomes at strain level compared to any of the other three binners (Fig. 1a and Supplementary Table 2). Interestingly, the increased performance of VAMB correlated (Pearson correlation coefficient = 0.90, linear regression $P = 0.035$) with the difficulty of the CAMI2 datasets, which we measured as the entropy of the genomes (Fig. 1b). Similarly, we found that VAMB reconstructed more genomes compared to MetaBAT2 at all levels of genome difficulty (Supplementary Fig. 5 and Supplementary Table 3). Additionally, we compared VAMB to ensemble binning, where bins from multiple programs are combined. Using DAS Tool[23], we tried combinations of the other binners and found VAMB to be better compared to all others (Fig. 1c). The addition of VAMB bins to DAS Tool improved the output of DAS Tool by up to 14% compared to VAMB only, but decreased performance on the Airways and MetaHIT datasets. VAMB and MetaBAT2 agreed on 39 NC genomes on average across the datasets and generated 49 and



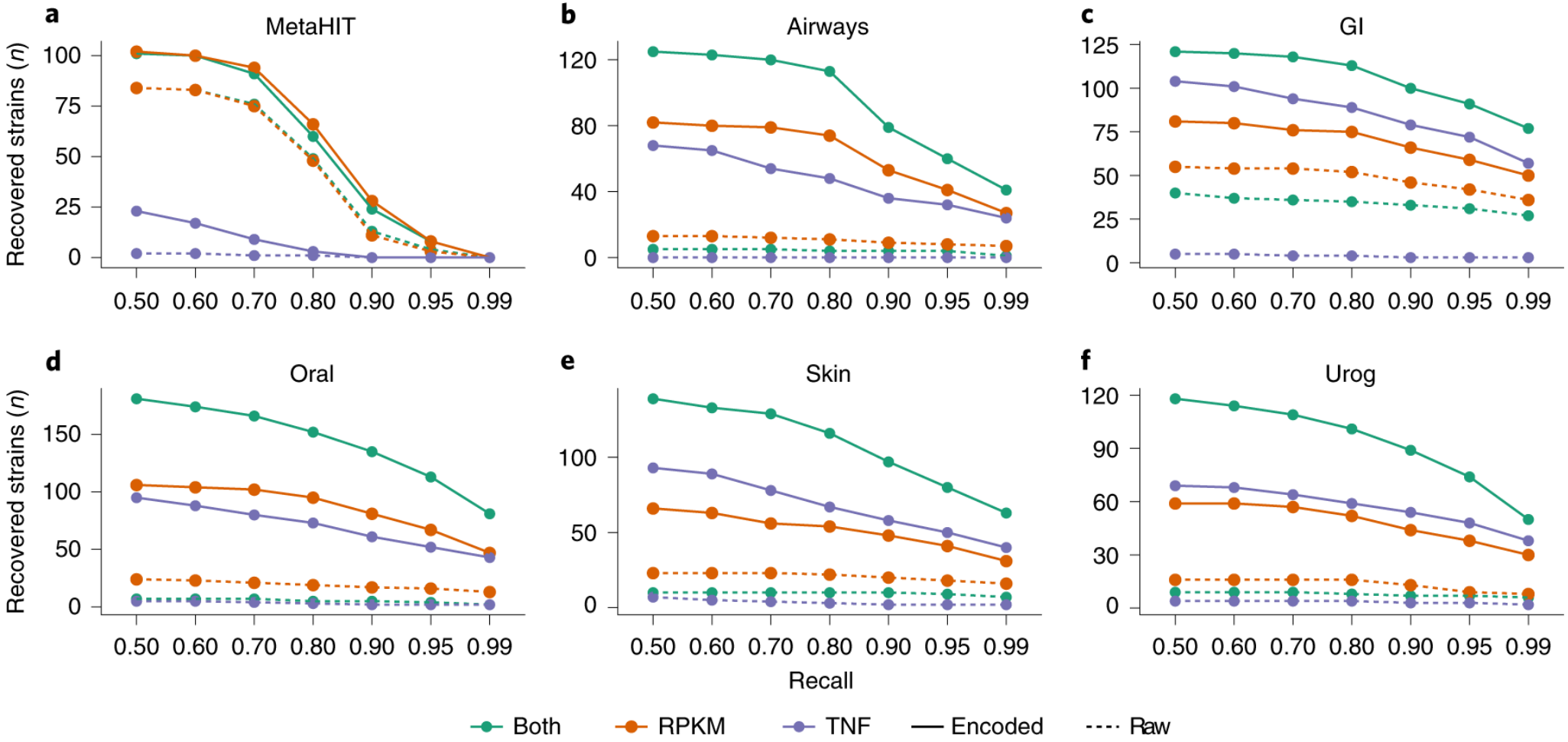






Variational Auto-encoders

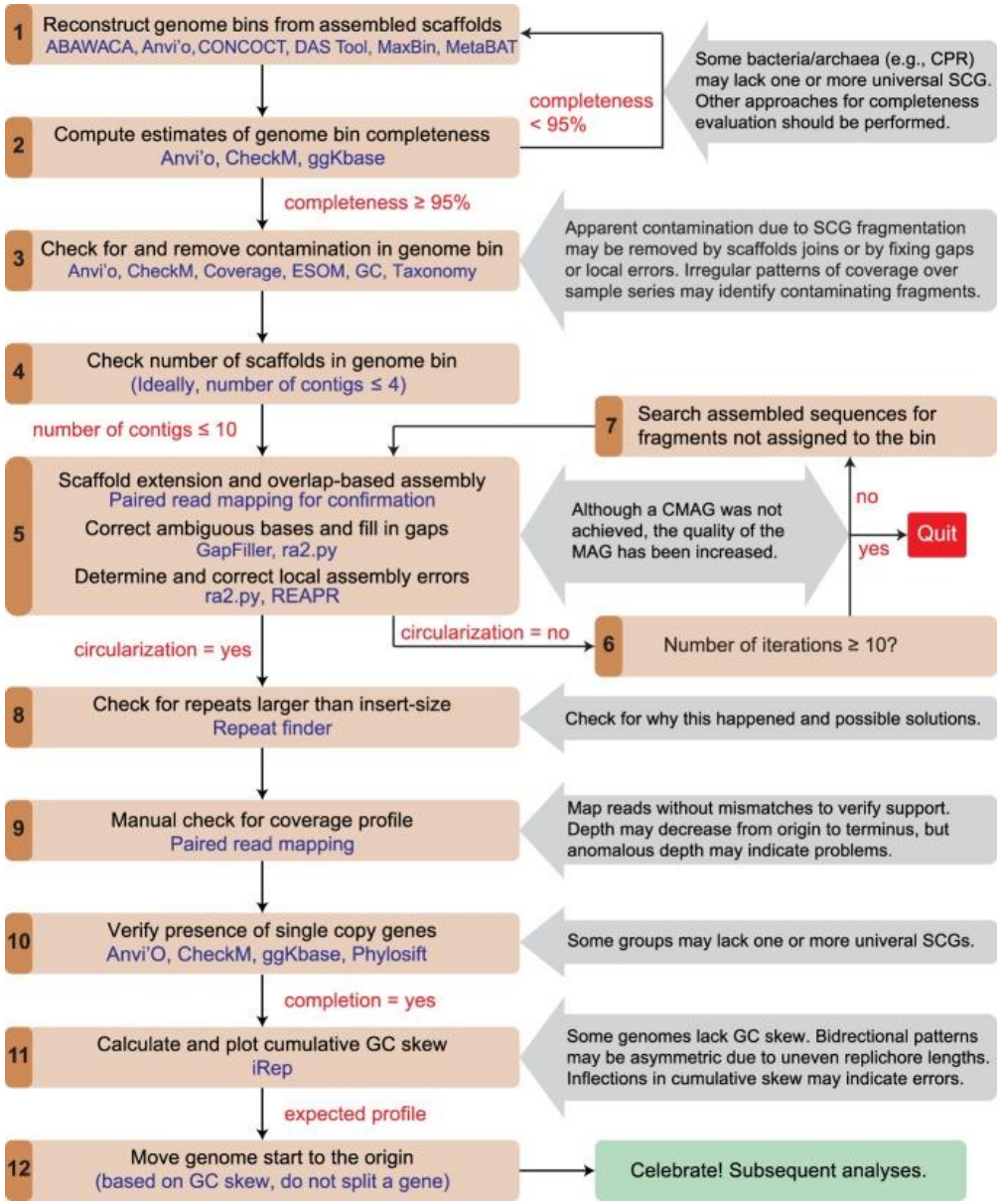
VAMB


a
MetaHIT
b
Airways
c
GI
d
Oral
e
Skin
f
Urog
Recovered strains (n)
Recall
0.50 0.60 0.70 0.80 0.90 0.95 0.99
Both
RPKM
TNF
Encoded
Raw

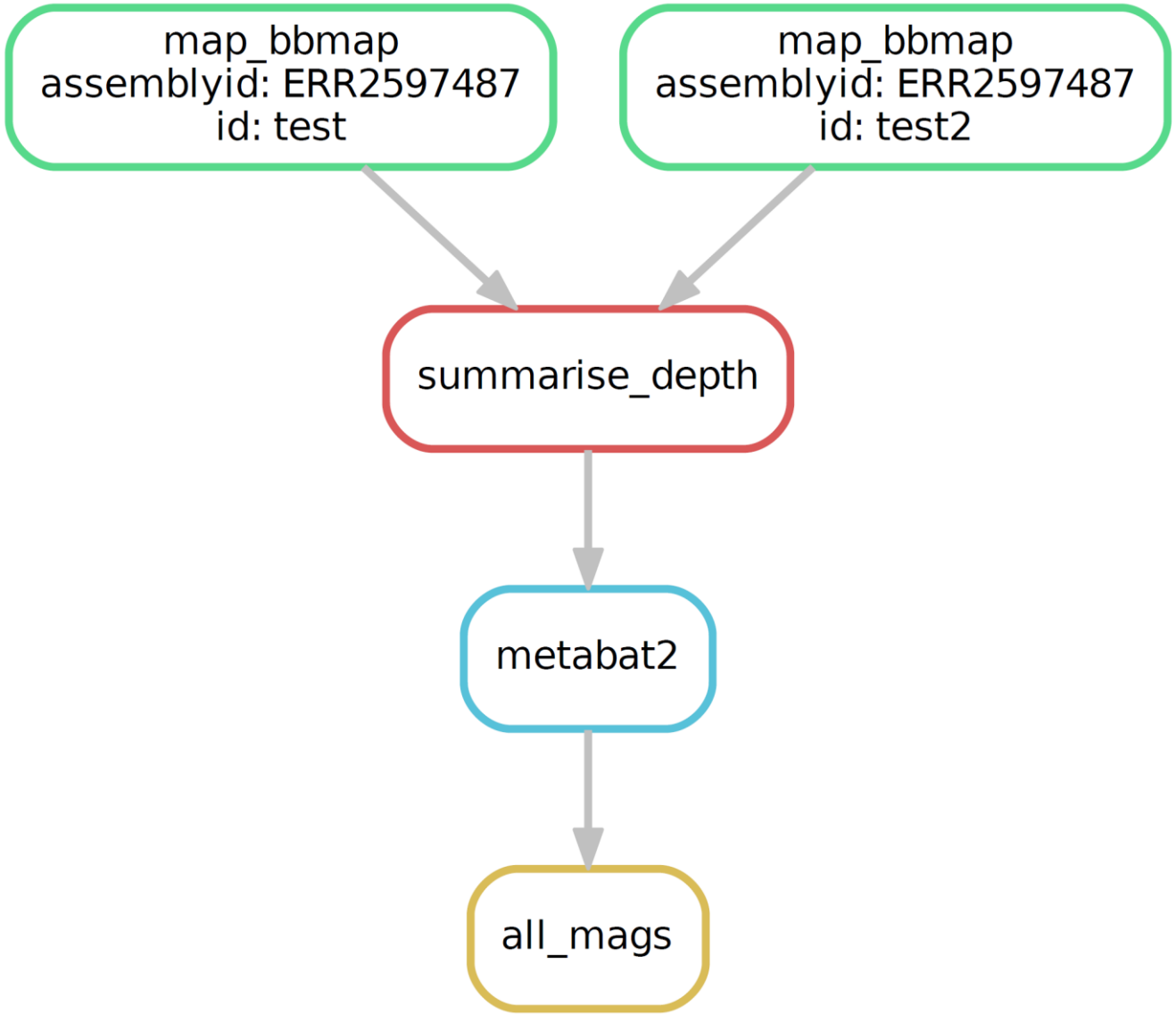
Potential bin refinement approaches

- Look for assembly errors and correct using PE read information
- Extract reads from binned scaffolds and re-assemble with single-genome assembler
- Overlap-assemble scaffolds that are binned together
- Attempt to circularize your sequence

Lin-Xing Chen et al, Genome Research (2020)



```
import os

localrules: map_bbmap, summarise_depth, metabat2

IDS, = glob_wildcards("../trim/{id}.singletons.fq.gz")

ASSEMBLYIDS, = glob_wildcards("../scaffolds_gt1000bp.{assemblyid}.fa")

# Combo rules
rule all_depths:
    input: expand("coverage/{assemblyid}.txt", assemblyid=ASSEMBLYIDS)

rule all_mags:
    input: expand("metabat2/{assemblyid}/{assemblyid}.bin", assemblyid=ASSEMBLYIDS)


# Map reads to assembly to get coverage and depth
rule map_bbmap:
    input:
        R1="../trim/{id}_1.trim.fq.gz",
        R2="../trim/{id}_2.trim.fq.gz",
        fa="../scaffolds_gt1000bp.{assemblyid}.fa"
    output:
        outsam="mapped/{assemblyid}_{id}.sam",
        outbam="mapped/{assemblyid}_{id}.sort.bam"
    log:
        out="logs/map_bbmap/{assemblyid}_{id}.out",
        err="logs/map_bbmap/{assemblyid}_{id}.err"

    threads: 20
    shell:
            '''
            module purge
            module load ngs tools
            module load java/1.8.0
            module load bbmap/38.90
            module load samtools/1.9
            mkdir -p logs/map_bbmap
            bbmap.sh in={input.R1} in2={input.R2} minid=0.90 threads={threads} ref={input.fa} outm={output.outsam} overwrite=t nodisk=t 2> {log.err} 1> {log.out}
            samtools view -bSh1 {output.outsam} | samtools sort -m 20G -@ 3 > {output.outbam}
            '''

rule summarise_depth:
    input:
        expand("mapped/{assemblyid}_{sample}.sort.bam", assemblyid=ASSEMBLYIDS, sample=IDS)
    output:
        dep="coverage/{assemblyid}.txt"
    log:
        out="logs/sum_depth/{assemblyid}.out",
        err="logs/sum_depth/{assemblyid}.err"
    params:
        bams="mapped/{assemblyid}_*.sort.bam"
    threads: 4
    shell:
            '''
            module purge
            module load ngs tools
            module load perl
            module load metabat/2.12.1
            mkdir -p logs/sum_depth
            #jgi_summarize_bam_contig_depths {input} --outputDepth {output.dep}
            jgi_summarize_bam_contig_depths {params.bams} --outputDepth {output.dep}
            '''

# Use MetaBat2 to bin scaffolds from sample
rule metabat2:
    input:
        asm="../scaffolds_gt1000bp.{assemblyid}.fa",
        dep="coverage/{assemblyid}.txt"
    output:
        "metabat2/{assemblyid}/{assemblyid}.bin"
    params:
        out="metabat2/{assemblyid}/{assemblyid}.bin",
        time="time/metabat2/{assemblyid}.time"
    log:
        out="logs/metabat2/{assemblyid}.out",
        err="logs/metabat2/{assemblyid}.err"
    benchmark:
        "benchmarks/{assemblyid}.metabat2.bm.txt"

    threads: 10
    shell:
            '''
            module purge
            module load ngs tools
            module load perl
            module load metabat/2.12.1
            mkdir -p logs/sum_depth
            jgi_summarize_bam_contig_depths {input} --outputDepth {output}
            mkdir -p logs/metabat2
            mkdir -p time/metabat2
            /usr/bin/time -v -o {params.time} metabat -t {threads} -v -m 1500 --saveCls -i {input.asm} -a {input.dep} -o {params.out} 2> {log.err} 1> {log.out}
            '''
```



Questions?