\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |
| --- |
| **https://babelfish.arc.nasa.gov/confluence/download/attachments/41354462/logo.JPG?version=1&modificationDate=1469045455180&api=v2** Bioinformatics Pipeline for Illumina Metagenomics Data  (Assembly-based analysis)  **Date: September 16, 2020 Revision:** -  **Document Number: GL-DPPD-XXXX**    meatball_white  **National Aeronautics and Space Administration**  Ames Research Center Moffett Field, CA 94035 |

Bioinformatics Pipeline for Illumina Metagenomics Data

(Assembly-based analysis)

Submitted by:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Michael D. Lee Date**

**GeneLab Science Team**

Approved by:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Sylvain Costes Date**

**GeneLab Project Manager**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Samrawit Gebre Date  
GeneLab Deputy Project Manager**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Homer Fogle Date  
GeneLab Data Processing Representative**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Jonathan Galazka Date  
GeneLab Project Scientist**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Anushree Sonic Date  
GeneLab Configuration Manager**

**CONFIGURATION MANAGEMENT**

This document is a GeneLab Configuration Management (CM)-controlled document. Changes to this document require approval of the GeneLab Project Manager. Changes to this document will be made by complete revision. Questions or comments concerning this document should be addressed to:

GeneLab Project Manager   
Mail Stop TBD  
Ames Research Center   
Moffett Field, California 94035

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **REV** | **DATE** | **DESCRIPTION** | **AUTHOR(S)** |
| - | 08/04/2020 | Initial release | Michael D. Lee |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Table of Contents**

[1.0 Overview 3](#_Toc530378852)

# Overview

The purpose of this document is to specify the software used and example code for processing Illumina shotgun metagenomics data towards a read-based analysis (i.e., no assembly). Raw data is generally the initial input of a Bioinformatics Pipeline, which is used to produce processed data. For any change to a Bioinformatics Pipeline that generates processed data, this completed document must be attached to the JIRA ticket requesting the public release of the generated processed data and is subject to a Change Control Board Review.

Type of omics data:rnaseq/microarray/proteomics/**metagenomics**/epigenomics/epitranscriptomics

Name of Bioinformatics Pipeline for proposed changes: Bioinformatics Pipeline for Illumina Metagenomics Data (Read-based analysis)

Justification for change request: Baseline

Affects to downstream processes (if any):

\*Attach an example of the proposed pipeline including all input and output files for each step affected by the change using a GeneLab dataset. Highlight the differences between the data generated using the current pipeline and the proposed pipeline.

**Step 1a: Raw Data QC**

|  |  |  |
| --- | --- | --- |
|  | Current | Proposed |
| Name of Software | FastQC |  |
| Version of software | v0.11.9 |  |
| Source of software (website is sufficient) | <https://www.bioinformatics.babraham.ac.uk/projects/fastqc/>  <https://anaconda.org/bioconda/fastqc> |  |
| Parameters Used / Example code: | fastqc -o raw\_fastqc\_output \*.fastq.gz  ## Parameter Definitions:   * `-o` – the output directory to store results * `\*.fastq.gz` – the input reads are specified as a positional argument, and can be all at once with wildcards like this, or as individual arguments with spaces in between them |  |
| Input data files | fastq, compressed or uncompressed (assumes human reads have been removed) |  |
| Output data files | fastqc.html (FastQC output html summary)  fastqc.zip (FastQC output data) |  |

**Step 1b: Compile Raw Data QC**

|  |  |  |
| --- | --- | --- |
|  | Current | Proposed |
| Name of Software | MultiQC |  |
| Version of software | v1.8 |  |
| Source of software (website is sufficient) | <https://multiqc.info/>  <https://anaconda.org/bioconda/multiqc> |  |
| Notes: | The `\` before line breaks can be ignored. They are only there to tell the computer to ignore the line-break characters that immediately follow them. |  |
| Parameters Used / Example code: | multiqc -o raw\_multiqc\_output \  raw\_fastqc\_output/  ## Parameter Definitions:   * `-o` – the output directory to store results * `raw\_fastqc\_output/` – the directory holding the output data from the fastqc run, provided as a positional argument |  |
| Input data files | fastqc.zip (FastQC output data) |  |
| Output data files | multiqc\_report.html (multiqc output html summary)  multiqc\_data (directory containing multiqc output data) |  |

**Step 2: Quality filter/trim raw reads**

|  |  |  |
| --- | --- | --- |
|  | Current | Proposed |
| Name of Software | bbduk (part of the bbtools/bbmap package) |  |
| Version of software | v38.86 |  |
| Source of software (website is sufficient) | <https://jgi.doe.gov/data-and-tools/bbtools/bb-tools-user-guide/bbduk-guide/>  <https://anaconda.org/bioconda/bbmap> |  |
| Parameters Used / Example code: | bbduk.sh in=sample-1-R1.fastq.gz \  in2=sample-1-R2.fastq.gz \  out1=sample-1-R1-trimmed.fastq.gz \  out2=sample-1-R2-trimmed.fastq.gz \  ref=ref-adapters.fa ktrim=l k=17 \  ftm=5 qtrim=rl trimq=10 mlf=0.5 \  maxns=0 > bbduk.log 2>&1  ## Parameter Definitions:   * `in` and `in2` – specifies the forward and reverse input reads, respectively * `out1` and `out2` – specifies the forward and reverse output reads, respectively * `ref` – specifies a fasta file holding potential adapter sequences (comes with bbduk installation) * `ktrim` – specifies to trim adapters from the 5’ end (left) if found * `k` – sets minimum length of kmer match to identify adapter sequences (provided by the “ref” file above) * `ftm` – sets a multiple of expected length the sequence should be (handles poor additional bases that are sometimes present, see [“Force-Trim Modulo” section on this page](https://jgi.doe.gov/data-and-tools/bbtools/bb-tools-user-guide/bbduk-guide/)) * `qtrim` – sets quality-score-based trimming to be applied to left and right sides * `trimq` – sets the score to use for PHRED-algorithm trimming * `mlf` – sets the minimum length of reads retained based on their initial length * `maxns` – sets the maximum number of Ns allowed in a read before it will be filtered out * `> bbduk.log 2>&1` – redirects the stderr and stdout (which is informative in this case) to a log file for saving |  |
| Input data files | fastq, compressed or uncompressed (original reads) |  |
| Output data files | fastq, compressed or uncompressed (trimmed reads)  tsv (per sample read counts before and after trimming)  txt (log file of standard output and error from bbduk run) |  |

**Step 3a: Trimmed Data QC**

|  |  |  |
| --- | --- | --- |
|  | Current | Proposed |
| Name of Software | FastQC |  |
| Version of software | v0.11.9 |  |
| Source of software (website is sufficient) | <https://www.bioinformatics.babraham.ac.uk/projects/fastqc/>  <https://anaconda.org/bioconda/fastqc> |  |
| Parameters Used / Example code: | fastqc -o trimmed\_fastqc\_output \  \*trimmed.fastq.gz  ## Parameter Definitions:   * `-o` – the output directory to store results * `\*trimmed.fastq.gz` – the input reads are specified as a positional argument, and can be all at once with wildcards like this, or as individual arguments with spaces in between them |  |
| Input data files | fastq, compressed or uncompressed (trimmed reads) |  |
| Output data files | fastqc.html (FastQC output html summary)  fastqc.zip (FastQC output data) |  |

**Step 3b: Compile Trimmed Data QC**

|  |  |  |
| --- | --- | --- |
|  | Current | Proposed |
| Name of Software | MultiQC |  |
| Version of software | v1.8 |  |
| Source of software (website is sufficient) | <https://multiqc.info/>  <https://anaconda.org/bioconda/multiqc> |  |
| Notes: | The `\` before line breaks can be ignored. They are only there to tell the computer to ignore the line-break characters that immediately follow them. |  |
| Parameters Used / Example code: | multiqc -o trimmed\_multiqc\_output \  trimmed\_fastqc\_output/  ## Parameter Definitions:   * `-o` – the output directory to store results * `trimmed\_fastqc\_output/` – the directory holding the output data from the fastqc run, provided as a positional argument |  |
| Input data files | fastqc.zip (FastQC output data) |  |
| Output data files | multiqc\_report.html (multiqc output html summary)  multiqc\_data (directory containing multiqc output data) |  |

**Step 4: Sample assembly**

|  |  |  |
| --- | --- | --- |
|  | Current | Proposed |
| Name of Software | megahit |  |
| Version of software | v1.2.9 |  |
| Source of software (website is sufficient) | <https://github.com/voutcn/megahit> |  |
| Notes: |  |  |
| Parameters Used / Example code: | megahit -1 sample-1-R1-trimmed.fastq.gz \  -2 sample-1-R2-trimmed.fastq.gz \  -o sample-1-assembly  -t 10 -m 500 > sample-1-assembly.log 2>&1  ## Parameter Definitions:   * `-1` and `-2` – the input forward and reverse reads * `-o` – the output directory * `-t` – number of threads * `-m` – minimum contig length to retain * `> sample-1-assembly.log 2>&1` – captures stdout/stderr to log file | ` |
| Input data files | fastq, compressed or uncompressed (quality-trimmed reads) |  |
| Output data files | fasta, assembly file  log, log file |  |

**Step 5: Renaming contigs and summarizing assemblies**

|  |  |  |
| --- | --- | --- |
|  | Current | Proposed |
| Name of Software | bit |  |
| Version of software | v1.8.11 |  |
| Source of software (website is sufficient) | <https://github.com/AstrobioMike/bioinf_tools>  <https://anaconda.org/AstrobioMike/bit> |  |
| Notes: |  |  |
| Parameters Used / Example code: | ## Renaming contigs so there won’t be any redundant names if multiple samples are involved  bit-rename-fasta-headers -i sample-1-assembly.tmp \  -w c\_sample-1 -o sample-1-assembly.fasta  ## Parameter Definitions:   * `i` – input fasta file * `-w` – wanted header prefix (a number will be appended), starts with a “c\_” to ensure they won’t start with a number which can be problematic * `-o` – output fasta file   ## Deleting intermediate file  rm \*-assembly.tmp  ## Summarizing assemblies  bit-summarize-assembly -o assembly-summaries.tsv \  \*assembly.fasta  ## Parameter Definitions:   * `-o` – output summary table * Multiple input assemblies can be provided as positional arguments | ` |
| Input data files | fasta, assembly file  fasta, contig-renamed fasta file (to summary program) |  |
| Output data files | fasta, contig-renamed fasta file  tsv, summary statistics of assemblies |  |

**Step 6: Gene prediction**

|  |  |  |
| --- | --- | --- |
|  | Current | Proposed |
| Name of Software | prodigal |  |
| Version of software | v2.6.3 |  |
| Source of software (website is sufficient) | <https://github.com/hyattpd/Prodigal> |  |
| Notes: |  |  |
| Parameters Used / Example code | prodigal -a sample-1-genes.faa \  -d sample-1-genes.fasta \  -f gff -p meta -c -q \  -o sample-1-gene-calls.gff \  -i sample-1-assembly.fasta  ## Parameter Definitions:   * `-a` – specifies the output amino acid sequences file * `-d` – specifies the output nucleotide sequences file * `-f` – specifies the output format gene-calls file * `-p` – specifies which mode run the gene-caller in * `-c` – no incomplete genes reported * `-q` – run in quiet mode (don’t output process on each contig) * `-o` – specifies the name of the output gene-calls file * `-i` – specifies the input assembly |  |
| Input data files | fasta, assembly fasta file |  |
| Output data files | fasta, sample-1-genes.faa; amino-acid fasta file  fasta, sample-1-genes.fasta; nucleotide fasta file  gff, sample-1-gene-calls.gff; gene-calls file in general feature format |  |

**Step 7: Functional annotation**

|  |  |  |
| --- | --- | --- |
|  | Current | Proposed |
| Name of Software | KOFamScan  bit |  |
| Version of software | KOFamScan: v1.3.0  bit: v1.8.10 |  |
| Source of software (website is sufficient) | <https://github.com/takaram/kofam_scan>  <https://www.genome.jp/kegg/>  <https://github.com/AstrobioMike/bioinf_tools>  <https://anaconda.org/AstrobioMike/bit> |  |
| Notes: | The annotation process overwrites the same temporary directory location if running multiple at the same time without specifying a unique one for each run. So it is good to specify a specific temporary directory for each sample with the `--tmp-dir` argument as shown below. |  |
| Parameters Used / Example code | ## downloading reference database of HMM models (only needs to be done once)  curl -LO ftp://ftp.genome.jp/pub/db/kofam/profiles.tar.gz  curl -LO ftp://ftp.genome.jp/pub/db/kofam/ko\_list.gz  tar -xzvf profiles.tar.gz  gunzip ko\_list.gz  ## Running KEGG annotation  exec\_annotation -p profiles/ -k ko\_list --cpu 15 \  -f detail-tsv -o sample-1-KO-tab.tmp \  --tmp-dir sample-1-tmp-KO \  --report-unannotated  sample-1-genes.faa  ## Parameter Definitions:   * `-p` – specifies the directory holding the downloaded reference HMMs * `-k` – specifies the downloaded reference KO terms * `--cpu` – specifies the number of searches to run in parallel * `-f` – specifies the output format * `-o` – specifies the output file name * `--tmp-dir` – specifies the temporary directory to write to (needed if running more than one concurrently, see Notes above) * `--report-unannotated` – specifies to generate an output for each entry * the input file is specified as a positional argument   ## Filtering output to retain only those passing the KO-specific score and top hits  bit-filter-KOFamScan-results -i sample-1-KO-tab.tmp \  -o sample-1-annotations.tsv  ## Parameter Definitions:   * `-i` – specifies the input table * `-o` – specifies the output table   ## Removing temporary files  rm -rf sample-1-tmp-KO/ sample-1-KO-annots.tmp |  |
| Input data files | fasta, gene-calls amino-acid fasta file |  |
| Output data files | tsv, table of KO annotations assigned to gene IDs |  |

**Step 8: Taxonomic classification**

|  |  |  |
| --- | --- | --- |
|  | Current | Proposed |
| Name of Software | CAT |  |
| Version of software | v5.1.2 |  |
| Source of software (website is sufficient) | <https://github.com/dutilh/CAT> |  |
| Notes: |  |  |
| Parameters Used / Example code | ## Pulling and un-packing their pre-built reference db (only needs to be done once)  wget tbb.bio.uu.nl/bastiaan/CAT\_prepare/CAT\_prepare\_20200618.tar.gz  tar -xvzf CAT\_prepare\_20200618.tar.gz  ## Running taxonomic classification  CAT contigs -c sample-1-assembly.fasta \  -d CAT-ref/2020-06-18\_database/ \  -t CAT-ref/2020-06-18\_taxonomy/ \  -p sample-1-genes.faa \  -o sample-1-tax-out.tmp \  -n 15 -r 3 --top 3 \  --I\_know\_what\_Im\_doing  ## Parameter Definitions:   * `-c` – specifies the input assembly fasta file * `-d` – specifies the CAT reference sequence database * `-t` – specifies the CAT reference taxonomy database * `-p` – specifies the input protein fasta file * `-o` – specifies the output prefix * `-n` – specifies the number of cores to use * `-r` – specifies the number of top protein hits to consider in assigning tax * `--top` – specifies the number of protein alignments to store * `--I\_know\_what\_Im\_doing` – allows us to alter the `--top` parameter   ## Adding taxonomy info from taxids to genes  CAT add\_names -i sample-1-tax-out.tmp.ORF2LCA.txt \  -o sample-1-gene-tax-out.tmp \  -t CAT-ref-taxonomy/ \  --only\_official  ## Adding taxonomy info from taxids to contigs  CAT add\_names -i sample-1-tax-out.tmp.contig2classification.txt \  -o sample-1-contig-tax-out.tmp \  -t CAT-ref/2020-06-18\_taxonomy/ \  --only\_official  ## Parameter Definitions:   * `-i` – specifies the input taxonomy file * `-o` – specifies the output file * `-t` – specifies the CAT reference taxonomy database * `--only\_official` – specifies to add only standard taxonomic ranks   ## Formatting gene-level output with awk and sed  awk -F $'\t' ' BEGIN { OFS=FS } \  { if ( $2 == "lineage" ) \  { print $1,$2,$4,$5,$6,$7,$8,$9,$10 } \  else if ( $2 == "ORF has no hit to database" \  || $2 ~ /^no taxid found/ ) \  { print $1,"NA","NA","NA","NA","NA","NA","NA","NA" } \  else { n=split($2,lineage,";"); \  print $1,lineage[n],$4,$5,$6,$7,$8,$9,$10 \  } } ' sample-1-gene-tax-out.tmp | \  sed 's/not classified/NA/g' | \  sed 's/superkingdom/domain/' | \  sed 's/^# ORF/gene\_ID/' | \  sed 's/lineage/taxid/' | \  sed 's/\\*//g' > sample-1-gene-tax-out.tsv  ## Formatting contig-level output with awk and sed  awk -F $'\t' ' BEGIN { OFS=FS } \  { if ( $2 == "classification" ) \  { print $1,$4,$6,$7,$8,$9,$10,$11,$12 } \  else if ( $2 == "unclassified" ) \  { print $1,"NA","NA","NA","NA","NA","NA","NA","NA" } \  else { n=split($4,lineage,";"); \  print $1,lineage[n],$6,$7,$8,$9,$10,$11,$12 \  } } ' sample-1-contig-tax-out.tmp | \  sed 's/not classified/NA/g' | \  sed 's/superkingdom/domain/' | \  sed 's/: [0-9\.]\*//g' | \  sed 's/^# contig/contig\_ID/' | \  sed 's/lineage/taxid/' | \  sed 's/\\*//g' > sample-1-contig-tax-out.tsv  ## Clearing intermediate files  rm sample-1\*.tmp\* |  |
| Input data files | fasta, assembly fasta file  fasta, gene-calls amino acid fasta file |  |
| Output data files | tsv, gene-level taxonomic classifications  tsv, contig-level taxonomic classifications |  |

**Step 9: Read-mapping**

|  |  |  |
| --- | --- | --- |
|  | Current | Proposed |
| Name of Software | bowtie2  samtools |  |
| Version of software | bowtie2: v2.3.5.1  samtools: v1.9 |  |
| Source of software (website is sufficient) | <https://github.com/BenLangmead/bowtie2>  <https://github.com/samtools/samtools> |  |
| Notes: |  |  |
| Parameters Used / Example code | ## Building reference index  bowtie2-build sample-1-assembly.fasta \  sample-1-assembly-bt-index  ## Parameter Definitions:   * first positional argument specifies the input assembly * second positional argument specifies the prefix of the output index files   ## Performing mapping, conversion to bam, and sorting  bowtie2 --threads 15 -x sample-1-assembly-bt-index \  -1 sample-1-R1-trimmed.fastq.gz \  -2 sample-1-R2-trimmed.fastq.gz \  | samtools view -b | \  samtools sort -@ 15 > sample-1.bam  ## Parameter Definitions:   * `--threads` – specifies the number of threads to run in parallel * `-x` – specifies the prefix of the reference index files to map to * `-1 and -2` – specifies the forward and reverse reads to map * `samtools view -b` – convert the output directly to bam format (compressed) * `samtools sort -@` – sort the bam file using 15 threads * `>` – redirect the output to a file   ## Indexing  samtools index -@ 15 sample-1.bam  ## Parameter Definitions:   * `-@` – set number of threads to use * input bam file is provided as a positional argument |  |
| Input data files | fasta, assembly file  fastq, trimmed read files |  |
| Output data files | bam, mapping file  bai, bam index file  log, read-mapping log file |  |

**Step 10: Getting coverage and detection information from bam files, and filtering coverages based on detection**

|  |  |  |
| --- | --- | --- |
|  | Current | Proposed |
| Name of Software | bbpileup (part of the bbtools/bbmap package) |  |
| Version of software | v38.86 |  |
| Source of software (website is sufficient) | <https://jgi.doe.gov/data-and-tools/bbtools/bb-tools-user-guide/bbduk-guide/>  <https://anaconda.org/bioconda/bbmap> |  |
| Notes: | “Detection” is a metric of what proportion of a reference sequence recruited reads (see here for visualization and example: <http://merenlab.org/2017/05/08/anvio-views/#detection>). Filtering coverage levels based on “detection” is one way of helping to mitigate non-specific read-recruitment. |  |
| Parameters Used / Example code | ## Generating coverage and detection info  pileup.sh -in sample-1.bam \  fastaorf=sample-1-genes.fasta \  outorf=sample-1-gene-cov-and-det.tmp \  out=sample-1-contig-cov-and-det.tmp  ## Parameter Definitions:   * `-in` – the input bam file * `fastaorf=` – input genes nucleotide fasta * `outorf=` – the output tsv file   ## Filtering gene coverage based on requiring 50% detection and parsing down to just gene ID and coverage  grep -v “#” sample-1-gene-cov-and-det.tmp | awk -F $’\t’ \  ‘ BEGIN { OFS=FS } { if ( $10 <= 0.5 ) $4 = 0 } \  { print $1,$4 } ‘ > sample-1-gene-cov.tmp  cat <( printf “gene\_ID\tcoverage\n” ) sample-1-gene-cov.tmp \  > sample-1-gene-coverages.tsv  ## Filtering contig coverage based on requiring 50% detection and parsing down to just contig ID and coverage  grep -v “#” sample-1-contig-cov-and-det.tmp | awk -F $’\t’ \  ‘ BEGIN { OFS=FS } { if ( $5 <= 50 ) $2 = 0 } \  { print $1,$2 } ‘ > sample-1-contig-cov.tmp  cat <( printf “contig\_ID\tcoverage\n” ) sample-1-contig-cov.tmp \  > sample-1-contig-coverages.tsv  ## Removing intermediate files  rm sample-1-\*.tmp |  |
| Input data files | bam, bam file  fasta, gene-calls nucleotide file |  |
| Output data files | tsv, table with gene-level coverages  tsv, table with contig-level coverages |  |

**Step 11: Combining gene coverages, taxonomy, and functional annotations into one table for each sample**

|  |  |  |
| --- | --- | --- |
|  | Current | Proposed |
| Name of Software | paste  awk  sed |  |
| Version of software | paste: v8.31  awk: v5.1.0  sed: v4.2.2 |  |
| Source of software (website is sufficient) | All are standard in any Unix-like environment: <https://www.gnu.org/software/> |  |
| Notes: |  |  |
| Parameters Used / Example code | paste <( tail -n +2 sample-1-gene-coverages.tsv | sort -V -k 1 ) \  <( tail -n +2 sample-1-annotations.tsv | sort -V -k 1 | cut -f 2- ) \  <( tail -n +2 sample-1-tax-out.tsv | sort -V -k 1 | cut -f 2- ) \  > sample-1-gene-tab.tmp  paste <( head -n 1 sample-1-gene-coverages.tsv ) \  <( head -n 1 sample-1-annotations.tsv | cut -f 2- ) \  <( head -n 1 sample-1-tax-out.tsv | cut -f 2- ) \  > sample-1-header.tmp  cat sample-1-header.tmp sample-1-gene-tab.tmp \  > sample-1-gene-coverage-annotation-and-tax.tsv  rm sample-1\*tmp sample-1-gene-coverages.tsv sample-1-annotations.tsv sample-1-gene-tax-out.tsv |  |
| Input data files | tsv, gene coverage, annotation, and taxonomy files |  |
| Output data files | tsv, table with combined gene coverage, annotation, and taxonomy info |  |

**Step 12: Combining contig coverages and taxonomy info into one table for each sample**

|  |  |  |
| --- | --- | --- |
|  | Current | Proposed |
| Name of Software | paste  awk  sed |  |
| Version of software | paste: v8.31  awk: v5.1.0  sed: v4.2.2 |  |
| Source of software (website is sufficient) | All are standard in any Unix-like environment: <https://www.gnu.org/software/> |  |
| Notes: |  |  |
| Parameters Used / Example code | paste <( tail -n +2 sample-1-contig-coverages.tsv | sort -V -k 1 ) \  <( tail -n +2 sample-1-contig-tax-out.tsv | sort -V -k 1 | cut -f 2- ) \  > sample-1-contig.tmp  paste <( head -n 1 sample-1-contig-coverages.tsv ) \  <( head -n 1 sample-1-contig-tax-out.tsv | cut -f 2- ) \  > 5492-contig-header.tmp  cat sample-1-contig-header.tmp sample-1-contig.tmp \  > sample-1-contig-coverage-annotation-and-tax.tsv  rm sample-1\*tmp sample-1-contig-coverages.tsv sample-1-contig-tax-out.tsv |  |
| Input data files | tsv, contig coverage and taxonomy files |  |
| Output data files | tsv, table with combined contig coverage and taxonomy info |  |

**Step 13: Generating normalized-coverage summary tables of KO-annotations and taxonomy across samples**

|  |  |  |
| --- | --- | --- |
|  | Current | Proposed |
| Name of Software | bit |  |
| Version of software | v1.8.10 |  |
| Source of software (website is sufficient) | <https://github.com/AstrobioMike/bioinf_tools>  <https://anaconda.org/AstrobioMike/bit> |  |
| Notes: |  |  |
| Parameters Used / Example code | bit-GL-combine-KO-and-tax-tables \*-gene-coverage-annotation-and-tax.tsv \  -o GLDS-286  ## Parameter Definitions:   * takes positional arguments specifying the input tsv files, can be provided as a space-delimited list of files, or with wildcards like above * `-o` – specify the output prefix (e.g. as above, will generate “GLDS-KO-function-coverages.tsv” and “GLDS-286-taxonomy-coverages.tsv” |  |
| Input data files | tsv, tables with gene-level coverage, functional annotations, and taxonomic classifications to combine |  |
| Output data files | tsv, table with all samples combined based on KO annotations (normalized to coverage per million genes covered)  tsv, table with all samples combined based on gene-level taxonomic classifications (normalized to coverage per million genes covered) |  |