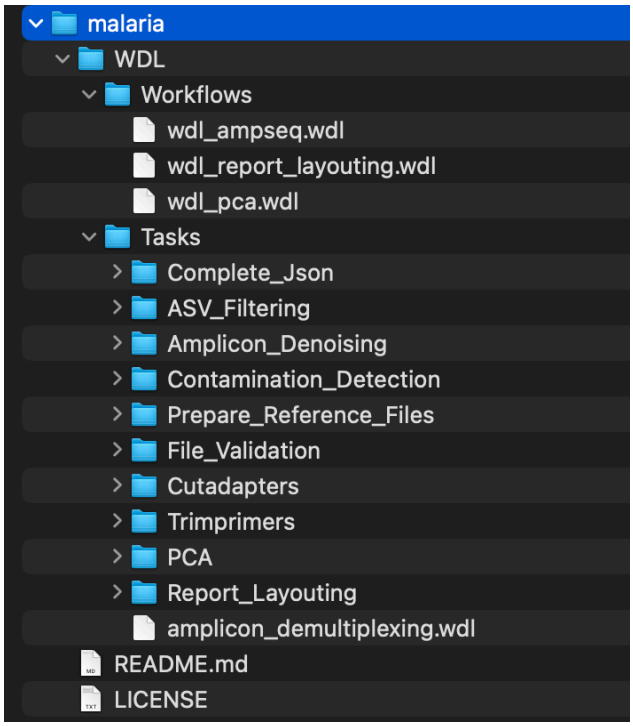
Malaria_Plasmodium_Illumina_Amplicon

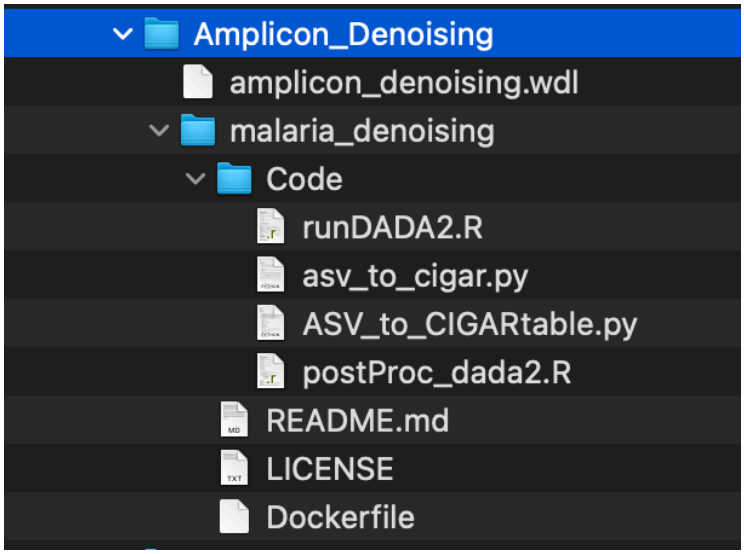
Development Guide

A workspace in Terra contains one or more workflows, each composed of a series of interconnected tasks. The *Malaria_Plasmodium_Illumina_Amplicon* GitHub repository is organized into nested directories, with a central directory named WDL. This directory includes two main subfolders: Workflows, which contains the workflow definitions, and Tasks, which houses the individual task definitions. Additionally, the WDL directory contains the README.md and LICENSE files, providing documentation and licensing information for the repository.



Each Tasks subdirectory contains a .wdl file with the task's instructions, along with an internal folder that includes a Dockerfile, README.md, and LICENSE. This internal folder may also contain a Code directory with proprietary code required by the task. This structure is consistent across all task directories.

For example, the Amplicon_Denoising task follows this layout.



Note: Some tasks, such as Trimprimers, do not include a Code directory when all necessary software is installed directly via the Dockerfile. In the case of Trimprimers, the task relies solely on Trim Galore, which is installed during the image build process.

Workflow structure

At the beginning of each workflow, the Tasks are Imported into the Workflow using the following command.

```
import "../Tasks/Prepare_Reference_Files/prepare_reference_files.wdl" as prepare_reference_files_t
```

For example, these are the Tasks, in the order they are called in the workflow, that compose wdl_ampseq (as of May 17th, 2025):

```
version 1.0
import "../Tasks/Prepare_Reference_Files/prepare_reference_files.wdl" as prepare_reference_files_t
import "../Tasks/File_Validation/validate_inputs.wdl" as validate_inputs_t
import "../Tasks/Contamination_Detection/contamination_detection.wdl" as contamination_detection_t
import "../Tasks/Cutadapters/cutadapters.wdl" as cutadapters_t
import "../Tasks/Trimprimers/trimprimers.wdl" as trimprimers_t
import "../Tasks/Amplicon_Denoising/amplicon_denoising.wdl" as amplicon_denoising_t
import "../Tasks/ASV_Filtering/asv_filtering.wdl" as asv_filtering_t
import "../Tasks/Complete_Json/complete_json.wdl" as complete_json_t
```

An Identifier is given to each Task when called in the WDL. For example:

call prepare_reference_files_t.prepare_reference_files as t_001_prepare_reference_files {
…

This can be seen clearer observing WDL/Worklfows/wdl_ampseq.wdl:
https://github.com/broadinstitute/malaria/blob/main/WDL/Workflows/wdl_ampseq.wdl

And one of the Tasks:
https://github.com/broadinstitute/malaria/blob/main/WDL/Tasks/Amplicon_Denoising/amplicon_denoising.wdl

All predefined variables specific to each task are declared within the task's input block. For instance, all parameters required to run DADA2 are defined in the Inputs section of amplicon_denoising.wdl, rather than being exposed in the main wdl_ampseq.wdl workflow.

This design helps keep the workflow file concise and improves readability by clearly associating each parameter with the task that uses it.

Additionally, note that variables defined at the task level are referenced within the command block using the tilde ~ syntax. In contrast, the dollar sign $ is reserved for referencing variables that are defined within the command <<< >>> block itself. This is a feature of WDL syntax.

Chaining tasks: How to use tasks outputs as the input of the next tasks

WDL documentation for additional reference:
https://docs.openwdl.org/en/latest/WDL/chain_tasks_together/

The outputs of one Task can be used as inputs for a subsequent task. For example, in wdl_ampseq.wdl, t_003_trimprimers uses t_002_cutadapters outputs as inputs:

```
call cutadapters_t.cutadapters as t_002_cutadapters {
  input:
    fastq1 = fastq1s[indx1],
    fastq2 = fastq2s[indx1]
}
call trimprimers_t.trimprimers as t_003_trimprimers {
  input:
```

```
        trimmed_fastq1 = t_002_cutadapters.fastq1_noadapters_o,
        trimmed_fastq2 = t_002_cutadapters.fastq2_noadapters_o
    }
```

Notice how the call naming standard facilitates associating the variables with their origins.

Dockerfiles

Docker containers are files that specify the contents of a virtual machine. They will vary depending on the task; however, some commonalities of all Dockerfiles used in this Workspace are listed below using the Dockerfile for the ASV_Filtering as an example:

```
FROM condaforge/mambaforge:23.3.1-0  #1. OPERATIVE  SYSTEM
RUN mamba config \                   #2. MAMBA INSTALLATIONS AND
    --add channels defaults \        #CHANNELS
    --add channels bioconda \
    --add channels conda-forge && \
    mamba create -n ampseq_env python=3.7.10 -y && \ #3. SOFTWARE
    mamba install -n ampseq_env \                    #INSTALLATION. NOTICE
    r-base=4.1 \                                        #THAT VERSIONING
    pandas=1.3.0 \                                      #USED
    biopython=1.79 \
    muscle=3.8.1551 \
    bioconductor-dada2=1.20.0 \
    bioconductor-limma=3.48.0 \
    bioconductor-biostrings=2.60.0 \
    r-data.table=1.14.0 \
    r-biocmanager=1.30.21 \
    r-ape=5.7_1 \
    r-vegan=2.6_4 \
    r-rcolorbrewer=1.1_3 \
    r-xlconnect=1.0.7 \
    r-data.table=1.14.0 \
    r-viridisLite=0.4.0 \
    r-argparse=2.0.3 \
    r-seqinr=4.2_5 \
    r-stringdist=0.9.8 \
    r-rmarkdown=2.22 \
    r-gridextra=2.3 \
    r-tidyverse=2.0.0 \
    -c conda-forge -c bioconda && \
    mamba clean --all -f -y && \
    echo "source activate ampseq_env" > ~/.bashrc
```

```
ENV PATH /opt/conda/envs/ampseq_env/bin:$PATH
ENV PATH /opt/conda/envs/ampseq_env/bin/python:$PATH
SHELL ["conda", "run", "-n", "ampseq_env", "/bin/bash", "-c"]
RUN apt-get update -y && apt-get install -y curl gnupg1 python3
RUN echo "deb [signed-by=/usr/share/keyrings/cloud.google.gpg]
http://packages.cloud.google.com/apt cloud-sdk main" | tee -a
/etc/apt/sources.list.d/google-cloud-sdk.list && curl
https://packages.cloud.google.com/apt/doc/apt-key.gpg | apt-key --keyring
/usr/share/keyrings/cloud.google.gpg  add - && apt-get update -y && apt-get install
google-cloud-sdk -y                          #4. GOOGLE INSTALLATION
RUN curl https://sdk.cloud.google.com | bash
ENV PATH=/root/google-cloud-sdk/bin/:${PATH}
RUN apt-get install -y gcc python3-dev python3-setuptools && pip3 uninstall -y crcmod
&& pip3 install --no-cache-dir -U crcmod
COPY Code Code                                    #5. CODE IMPORTATION
```

Testing and Debugging Protocol

Testing and debugging WDL tasks is an iterative process. The recommended workflow is as follows:

1. **Modify the** Dockerfile **and Code Directory**

Make any necessary changes to the Dockerfile and the associated Code/ directory inside the task folder.

2. **Build and Push the Docker Image**

Use a standard command such as:

docker build --platform linux/amd64 -t jorgeamaya/fileprep_ampseq:v_0_0_3 . && docker push jorgeamaya/fileprep_ampseq:v_0_0_3

Note: Always ensure the Docker image tag (v_0_0_3 in this example) matches the version specified in the WDL task's runtime block. During development, use a version that differs from the stable release to avoid overwriting production containers.

3. **Update the Task WDL (if needed)**

Modify the .wdl file for the task to reflect changes in the Docker image or code structure. This may include updating runtime attributes to point to the new Docker image version.

4. **Run the Workflow Locally**

For testing, it's common to use a dedicated "Tester" repository containing a small dataset. Repositories for various panels can be found here.
To execute the workflow locally using Cromwell, run:

rm -rf cromwell-* | cromwell run ~/Desktop/malaria/WDL/Workflows/wdl_ampseq.wdl -i test.json

Note:

- Update the path to the malaria repository to match the local file system structure.
- The test.json file should contain the minimal set of inputs required to successfully run the workflow.

The following is an example of a minimal test.json file (as of May 17th, 2025).

```
{
  "ampseq.fastq1s": [
    "path/Tester_Repo_Hackathon/SENEGAL_Test/DC3-6_S43_L001_R1_001.fastq.gz",
    "path/Tester_Repo_Hackathon/SENEGAL_Test/DC3-7_S89_L001_R1_001.fastq.gz",
"path/Tester_Repo_Hackathon/SENEGAL_Test/SN23DBL032_S92_L001_R1_001.fastq.gz"
,
"path/Tester_Repo_Hackathon/SENEGAL_Test/SN23DBL041_S91_L001_R1_001.fastq.gz"
  ],
  "ampseq.fastq2s": [
    "path/Tester_Repo_Hackathon/SENEGAL_Test/DC3-6_S43_L001_R2_001.fastq.gz",
    "path/Tester_Repo_Hackathon/SENEGAL_Test/DC3-7_S89_L001_R2_001.fastq.gz",
"path/Tester_Repo_Hackathon/SENEGAL_Test/SN23DBL032_S92_L001_R2_001.fastq.gz"
,
"path/Tester_Repo_Hackathon/SENEGAL_Test/SN23DBL041_S91_L001_R2_001.fastq.gz"
  ],
  "ampseq.reference_genome":
"path/Tester_Repo_Hackathon/SENEGAL_Test/PkPfPmPoPv.fasta",
  "ampseq.panel_info":
"path/Tester_Repo_Hackathon/SENEGAL_Test/MAD4HatTer_info.tsv",
  "ampseq.sample_metadata":
"path/Tester_Repo_Hackathon/SENEGAL_Test/Sample_Metadata.tsv"
}
```

5. Evaluating Results

Each time a Cromwell process is executed, a new cromwell-executions directory is created. This directory stores the outputs of each task, along with detailed runtime information. To diagnose errors and evaluate task behavior, developers should examine the following files within each task's subdirectory:

- stdout — captures standard output messages from the task
- stderr — captures standard error messages and is often the first place to look when debugging
- Any other task-specific output files

Inspection of these files is necessary for understanding failures or unexpected results during workflow development and testing.

For example, running the pipeline will produce following directories within cromwell-executions. Each directory contains the output of one task.

ls cromwell-executions/ampseq/29486805-ade1-4868-8d8c-81c02b8ab2b2/call-t_00*
call-t_000_prepare_reference_files/ call-t_003_trimprimers/
call-t_006_complete_json/
call-t_001_validate_fastqs/ call-t_004_amplicon_denoising/
call-t_002_cutadapters/ call-t_005_asv_filtering/

Looking inside call-t_004_amplicon_denoising/ will show stdout, stderr, and the Results directory.

ls
cromwell-executions/ampseq/29486805-ade1-4868-8d8c-81c02b8ab2b2/call-t_004_amplicon_denoising/execution
Results samples_tmp.csv script.submit stdout
docker_cid script stderr stdout.background
rc script.background stderr.background

Files

User must provide:

1. Paired-end FASTQ files. The files must follow the Illumina's standard naming. For example (parts in red are mandatory):

- 1000-11-MIX-R1_S1_L001_R1_001.fastq.gz
- 1000-11-MIX-R1_S1_L001_R2_001.fastq.gz
- 1000-11-MIX-R2_S2_L001_R1_001.fastq.gz
- 1000-11-MIX-R2_S2_L001_R2_001.fastq.gz

2. A CSV metadata file. The file must contain, as a minimum, the following columns.

   Sample_id,Country,Latitude,Longitude
   1000-11-MIX-R1_S1,Placeholder,0,0
   1000-11-MIX-R2_S2,Placeholder,0,0
   1000-11-MIX-R3_S3,Placeholder,0,0
   1000-11-MIX-R4_S136,Placeholder,0,0

   Notice that Placeholder and 0s may be placed in case information is missing. We advise using the country's centroid whenever information is missing.

Parameters

Task name	Variable	Type	Input value
ampseq	fastq1s	Array[File]	Forward fastq.gz files to run the pipeline. Must be provided as an array, not as individual files.
ampseq	fastq2s	Array[File]	Reverse fastq.gz files to run the pipeline. Must be provided as an array, not as individual files.
ampseq	panel_info	File	File with panel information.
ampseq	reference_genome	File	Fasta file with the reference genome to be used in the pipeline.
ampseq	sample_metadata	File	CSV metadata file
ampseq	barcodes_index	File	File with barcodes index information. If not provided, the pipeline will not use barcodes.
ampseq	forward_primers_file	File	File with forward primers with inline barcodes. Optional, if not provided the pipeline will not use barcodes.
ampseq	path_to_snv	File	Path to the SNV file. Optional, if not provided the pipeline will not remove SNVs provided in the SNV file.
ampseq	reference_amplicons	File	Fasta file with the reference amplicons. This is optional, if not provided the pipeline automatically generate reference amplicons.
ampseq	reference_amplicons_2	File	Fasta file with the second set of reference amplicons. This is optional, if not provided the pipeline will use only one set of reference amplicons.
ampseq	reverse_primers_file	File	File with reverse primers with inline barcodes. Optional
ampseq	sample_ids	Array[String]	Array of sample IDs. If not provided, the pipeline will use the file names of the fastq files as sample IDs.

t_0001_contamination_detection	contamination_threshold	Int	Minimum number of reads to consider a sample contaminated. Default is 0.
t_0001_contamination_detection	minreads_threshold	Int	Minimum number of reads to consider that a sample has too few reads. Default is 1000.
t_0001_contamination_detection	primer_distance_threshold	Int	Maximum Hamming distance between primers template and detected primer to accept the primer. Default is 2.
t_001_validate_fastqs	use_validated_fastqs	Boolean	If true, the pipeline will use the validated fastq files. If false, the pipeline will use the original fastq files.
t_002_cutadapters	downsample_fraction	Int	Extracts a fraction of the reads from the input fastq files. This is useful for processing large datasets.
t_002_cutadapters	trim_galore_length	Int	Discard reads shorter than this value after trimming. Default is 20.
t_002_cutadapters	trim_galore_qvalue	Int	Specifies the minimum quality value (Phred score) to keep a base. Default is 5.
t_004_amplicon_denoising	Class	String	Name of the column in the metadata file for sample class
t_004_amplicon_denoising	exclude_bimeras	Boolean	Exclude bimeras from analysis
t_004_amplicon_denoising	include_failed	Boolean	Include failed samples in the analysis
t_004_amplicon_denoising	justConcatenate	Int	Whether to just concatenate reads without merging
t_004_amplicon_denoising	matchIDs	String	Whether to match IDs on fastqs
t_004_amplicon_denoising	max_consist	Int	Maximum number of mismatches in overlap region
t_004_amplicon_denoising	max_indel_dist	String	Maximum distance for indels
t_004_amplicon_denoising	max_snv_dist	String	Maximum distance for SNVs
t_004_amplicon_denoising	maxEE	String	Maximum expected error rate
t_004_amplicon_denoising	maxMismatch	Int	Maximum number of mismatches allowed during merging
t_004_amplicon_denoising	min_reads	String	Minimum number of reads required for a sample to be included in the analysis
t_004_amplicon_denoising	min_samples	String	Minimum number of samples required for the analysis
t_004_amplicon_denoising	minLen	Int	Minimum length of reads to retain
t_004_amplicon_denoising	omegaA	Float	Alpha parameter for consensus quality score
t_004_amplicon_denoising	polyN	String	Minimum homopolymer length
t_004_amplicon_denoising	pool	String	Whether to pool samples for denoising
t_004_amplicon_denoising	saveRdata	String	Whether to save the intermediate R data files
t_004_amplicon_denoising	strain	String	Plasmodium strain to be used in the analysis

t_004_amplicon_denoising	strain2	String	Second Plasmodium strain to be used in the analysis
t_004_amplicon_denoising	trimRight	String	Number of bases to trim from the 3' end
t_004_amplicon_denoising	truncQ	String	Quality threshold for truncating reads
t_005_asv_filtering	ampseq_export_format	String	Format for exporting ASVs tables.
t_005_asv_filtering	bimera_formula	String	Formula to detect bimeras.
t_005_asv_filtering	flanking_INDEL_formula	String	Formula to mask INDELs in the homopolymeric regions of ASVs
t_005_asv_filtering	homopolymer_length	Int	Formula to mask SNVs in the homopolymeric regions of ASVs
t_005_asv_filtering	INDEL_in_homopolymer_formula	String	Formula to mask INDELs in the homopolymeric regions of ASVs
t_005_asv_filtering	locus_ampl_rate	Float	Minimum proportion of amplified samples at a locus required to keep a locus
t_005_asv_filtering	metadata_latitude_name	String	Name of the column in the metadata file containing latitude information
t_005_asv_filtering	metadata_longitude_name	String	Name of the column in the metadata file containing longitude information
t_005_asv_filtering	metadata_variable1_name	String	Name of the column with categories to separate samples
t_005_asv_filtering	metadata_variable2_name	String	Name of the second column with categories to separate samples
t_005_asv_filtering	min_abd	Int	Integer that define the minimum read depth of trusted alleles. Alleles bellow this threshold will be removed. The default is 10 reads, however, if data was generated by an iSeq100 machine or if the sequencing run was overloaded with samples (high occupancy and low QC30) a lower threshold is recommended.
t_005_asv_filtering	min_ratio	Float	Numeric value that define the minimum ratio between the read depth of minor alleles respect to major alleles in heterozygous positions. Minor alleles bellow this threshold will be discarded. (default 0.1) As for the previous parameter, the ratio depends on the quality and depth of the sequencing run.
t_005_asv_filtering	off_target_formula	String	Formula used to identify and remove off-targget PCR products. By default the parameter density of variant sites of the ASV (or allele i) per amplicon j (dVSTIES_ij) is used, but other metrics can be employed also
t_005_asv_filtering	PCR_errors_formula	String	Formula used to define other kinds of PCR errors.
t_005_asv_filtering	sample_ampl_rate	Float	Min proportion of amplified loci by a sample that is required for the sample to be kept (default 0.75)
t_005_asv_filtering	sample_id_pat	String	Regular expression the allows to identify samples of interest and remove controls or

			other undesired samples that we don,Äôt want in further analysis
t_005_asv_filtering	SNV_in_homopolymer_formula	String	Formula for masking SNVs in homopolymer regions of ASVs

The identification of all these likely genotyping errors is based on the following parameters:

1. dVSITES_ij: Density of variant sites (SNPs and INDELs) in the allele (ASV) j of the locus i.
2. P_ij ($P_{i,j}$): The total number of samples that amplified each alternative allele j in locus i.
3. H_ij ($H_{i,j}$): Number of heterozygous samples in the locus i that carry the alternative allele j.
4. H_ijminor ($H_{i,jminor}$): Number of heterozygous samples in the locus i where the alternative allele j is the minor Allele (the allele with the lower read counts).
5. h_ij ($h_{i,j} = H_{i,j}/P_{i,j}$): ratio of heterozygous samples in the locus that carry the alternative allele of interest respect to the total number of samples that amplified alternative allele.
6. h_ijminor ($h_{i,jminor} = H_{i,jminor}/H_{i,j}$): ratio of heterozygous samples where the alternative allele is the minor Allele respect to the total number of heterozygous samples in the locus that carry the alternative allele.
7. p_ij ($p_{i,j}$): population prevalence of the alternative allele j in locus i.
8. flanking_INDEL: Boolean that specify if there are INDELs in the flanking region of the ASV.
9. SNV_in_homopolymer: Boolean that specify if there are SNVs in an homopolymer region. By default and homopolymer is defined as a region of a single nucleotide repeated 5 times in tandem.
10. INDEL_in_homopolymer: Boolean that specify if there are SNVs in an homopolymer region. By default and homopolymer is defined as a region of a single nucleotide repeated 5 times in tandem.