MIPTools

Bailey Lab

May 16, 2024

QUICK START

Welcome to the MIPTools User Guide!

MIPTools is a suite of computational tools that are used for molecular inversion probe (MIP) design, data processing, and analysis. Throughout much of this tutorial, we assume a user interested in using MIPs as a cost effective way to amplify and sequence hundreds to thousands of targeted regions of the genomes from hundreds to thousands of pooled barcoded samples. Our group primarily uses these MIPs for genomic surveillance of Plasmodium falciparum targets, but we have attempted to generalize this tool for other questions and datasets. This toolset also assumes the use of unique molecular identifiers (UMIs) which tell us how much of each type of DNA was originally present in each sample (prior to PCR amplification).

A typical pipeline might look something like this:

- First, a user might design MIP probes (using the probe design tool of this program) that have UMIs added to each MIP probe.
- Second, a user might perform mip capturing reactions, PCR, sample barcoding, and illumina sequencing. The output data should be demultiplexed, resulting in two fastq files per sample. Bench techniques for these experiments are described elsewhere.
- Third, the data is wrangled to generate an output file describing which genotypes (or haplotypes) are found at which abundances in each sample for each targeted region, using:
  - a sample sheet that describes the samples
  - a fastq folder of samples
  - a project resources folder that describes the probes
- Finally, the haplotype data is analyzed using a variant caller (Freebayes is currently our best-supported tool) to produce a VCF file and some output tables with frequencies and prevalences of mutations of interest, using:
  - a sample sheet that describes the samples
  - a folder containing the wrangled haplotype data
  - a folder of indexed genomes for your species of interest
  - a project resources folder that describes the probes

CHAPTER

ONE

INSTALLATION

If you have a working copy of singularity and have demultiplexed data, you can use our sif file (recommended approach, below). Due to licensing issues, if you want to use miptools to demultiplex your data (we're just wrapping up illumina's bcl2fastq demultiplexing program) you'll need to download bcl2fastq, put it in our 'programs' folder, and build your own sif file from source.

1.1 Dependencies

A working copy of Singularity is required. Our tools only work when singularity is installed with administrative privileges with **sudo**. Administrative privileges are not required to run singularity.

Singularity is available for most Linux systems and is usually already installed on academic high-performance clusters. It is also possible to install and use on Mac OS using virtual machines with a little bit of extra work.

1.2 Easy Installation

The MIPTools container, built and ready to use, can be downloaded . You can download the development version or any previous release:

```
# Download the latest stable release
wget https://baileylab.brown.edu/MIPTools/download/miptools_v0.4.0.sif

# Download the development version
wget https://baileylab.brown.edu/MIPTools/download/miptools_dev.sif
```

1.3 Install From Source

MIPTools can also be built from source using the definition file provided in the GitHub repository. You can install the most recent release using the following:

```
# Install stable version
git clone --branch v0.4.0 https://github.com/bailey-lab/MIPTools.git
```

You can alternatively install the development version:

```
# Install dev version
git clone https://github.com/bailey-lab/MIPTools.git
```

Next, simply build the container and you should be all set to get started using MIPTools!

```
cd MIPTools
sudo singularity build miptools.sif MIPTools.def
```

`miptools.sif` is a single **portable** file which has all the programs needed for MIP design, data analysis, and a lot more.

1.3.1 Sudo Privileges

Warning: You must have `sudo` privileges to *build* the image. You do not need `sudo` to *use* the image.

If you want to run the container on an environment without `sudo`, either download a prebuilt image (see above) or build the container on your own machine where you *do* have `sudo` privilege and copy the image file to the computer without `sudo`. Note that the Singularity program itself must have been installed with `sudo`.

1.3.2 Demultiplexing

If you plan to use MIPTools to demultiplex bcl files, you must download `bcl2fastq` separately. Currently, you can download it from here. You must download the file: `bcl2fastq2 Conversion Software v2.20 Installer (Linux rpm)` and place it in the `MIPTools/programs` directory.

1.3.3 CPU Usage

The build process can take about 30-60 minutes to build. By default, the build process will use 20 CPU cores. You can change this by editing the `CPU_COUNT=20` value at the top of the `MIPTools.def` file to a suitable number before building the container.

CHAPTER

TWO

ABBREVIATED TUTORIAL

This abbreviated tutorial provides a minimal version of instructions for people who want to get a "working version" up and running quickly to explore the data for themselves. A little background is still needed.

2.1 Background

MIPTools supports several computational steps, including:

- MIP design: This is for choosing resions of the genome that you'd like to target.
- Wrangling: This is for finding what haplotypes are associated with each targeted region and the number of times each haplotype was seen in each sample. This can be thought of as the "core" purpose of MIPTools. The output is a tab delimited file called **allInfo.tsv.gz**
- stat-checking: This is for figuring out which samples and which targeted regions (aka MIPs) performed well and which did not. There are several graphical outputs and comma separated files produced at this stage. Among the most important are **barcode_counts.csv** (how many times each targeted region of the genome was seen in each sample) and **repool.csv** (which MIPs need to be re-sequenced or repooled in each sample). **umi_heatmap.html** is also useful for visualizing the performance of each MIP in each sample (open this with a web browser).
- variant calling: This is for estimating the number of times that each MIP was associated with a mutation in each sample. The outputs are a VCF file containing mutated genomic positions for each sample and several tables that annotate mutations with associated amino acid changes for each sample. For the tutorial, the main outputs of interest are three tables that can be used to infer which samples contain each mutation:
  - *coverage_AA_table.csv*: how many times the mutation was sequenced
  - *reference_AA_table.csv*: how many times the reference allele was seen in each sample
  - *alternate_AA_table.csv*: how many times the alternate (mutant) allele was seen in each sample

2.1.1 Input File Structure

A few directories are required for most operations.

- **species_resources:** Contains information about the genome you targeted MIPs against. Bound internally to `/opt/species_resources`. Includes:
  - *fasta file*: Genome reference sequence in fasta format.
  - *bowtie2_genome*: The reference genome indexed using bowtie2.
  - *bwa_genome*: The reference genome indexed using bwa.

  - *SNPs*: VCF formatted locations of known SNPs in the reference genome. Useful during MIP design to avoid targeting a polymorphic region of the genome.
  - *refgene*: RefGen style gene/gene prediction table in GenePred format. These are available at http://genome.ucsc.edu under Tools/Table Browser for most species. If you have gff3/gtf formatted files, they can be converted to GenePred format using Jim Kent's programs gff3ToGenePred and gtfToGenePred.
  - *refgene_tabix*: An index of the refgene file, created using tabix.
  - *file_locations.tsv*: This file is required for all operations. It is a tab separated text file showing where each required file will be located in the container. Each line corresponds to one file. First field states the species for the file, second field states what kind of file it is and the last field is the absolute path to the file within the singularity container

- **project_resources:** Contains information about your mip panel. Bound internally to `/opt/project_resources`. Includes:
  - A targets.tsv file with the genomic coordinates of any protein-coding mutations that are of particular interest.
  - a mip_ids folder that contains the mip arms of MIPs that target regions of the genome that are of interest.
  - A few redundant json and csv files for easy access to MIP information. Our goal is to remove these in the future.

2.2 Analyzing the Data

Now that we know what steps will be performed and how files are organized, we can look at a hypothetical dataset. The tutorial dataset contains 56 samples sequenced with 57 targeted genomic regions corresponding to known drug resistance mutations of the P. falciparum genome. It assumes that MIPs have already been designed, and that these MIPs have been used to target our regions of interest, and that samples have already been pooled together, sequenced with illumina paired end reads, and demultiplexed.

You can download the tutorial dataset from here:
https://baileylab.brown.edu/MIPTools/download/test-data.tar.gz
The dataset includes a project_resources folder, a species_resources folder, a sample sheet, and a fastq directory with demultiplexed illumina paired end reads as output.

You can obtain a copy of our latest sif file from here:
https://baileylab.brown.edu/MIPTools/download/miptools_dev.sif
This includes all executable programs needed for analysis

2.2.1 Wrangling

For convenience, settings can be passed in to the wrangler via a yaml file. Later in the tutorial, we'll show some more advanced usage options available for troubleshooting and passing more customizable inputs. For now, you can obtain an example simple settings file for wrangling with this command:

```
wget
https://github.com/bailey-lab/MIPTools/raw/master/user_scripts/wrangler_by_sample.yaml
```

After downloading, open the file for editing with a text editor and make sure to **follow the instructions in this file**, editing it to contain the correct path to the project resources, species resources, sample sheet, and sif files you downloaded above, as well as the location where you'd like the output to be sent.

We've also provided a bash script for converting the yaml settings into instructions for the wrangler. You can obtain the bash script for wrangling with this command (put it in the same folder as the settings yaml file):

```
wget https://github.com/bailey-lab/MIPTools/raw/master/user_scripts/wrangler_by_sample.sh
```

After changing directory to a folder that can run your data, you can execute the wrangler script with:

```
bash wrangler_by_sample.sh
```

2.2.2 Checking run stats

After wrangling is finished, you can obtain a settings file for checking run stats with this command:

```
wget https://github.com/bailey-lab/MIPTools/raw/master/user_scripts/variant_calling.yaml
```

Make sure to follow the instructions in this file.

You can obtain the script for checking run stats here (put it in the same folder as the settings file):

```
wget https://github.com/bailey-lab/MIPTools/raw/master/user_scripts/check_run_stats.sh
```

And you can execute it like this:

```
bash check_run_stats.sh
```

2.2.3 Variant Calling

Variant calling uses the same settings file as check_run_stats.

You can obtain the script for variant calling here (put it in the same folder as the settings file):

```
wget https://github.com/bailey-lab/MIPTools/raw/master/user_scripts/variant_calling.sh
```

And you can execute it like this:

```
bash variant_calling.sh
```

2.3 Resource Requirements

If you use the default processor counts, wrangling and variant calling should complete in approximately five minutes each for the tutorial dataset, with checking run stats completing substantially faster.

More generally, resources required vary widely depending on the project. Wrangling and variant calling require the most RAM and processing power, and both of these steps can be parallelized across multiple processors. Wrangling with ~7,000 samples can take up to three days to complete, and some variant calling steps on datasets this large can take a little over a week. The more processors (also known as CPUs or threads) you ask for, the faster the job will run, the more RAM will be required, and the higher the probability that the job will crash if RAM is insufficient. With a dataset containing 7,000 samples, a single processor might require up to 150 GB of RAM in the variant calling step. Internally, MIPTools uses snakemake so that if a job crashes partway through, you can rerun it and MIPTools will pick up where it left off. Therefore, you might consider running a job once and requesting a large number of processors (e.g. 15) so that most of the steps finish quickly. Then, if the job crashes, you might edit the settings file to request fewer

processors (e.g. 4 or even 2 or 1) so that any remaining particularly tricky steps can be run with a lower likelihood of crashing.

CHAPTER

THREE

PROBE DESIGN

This guide provides a walkthrough of how to design molecular inversion probes (MIPs) using MIPTools. We use a test data set to demonstrate each step in the design process.

3.1 Download Test Data

The test data set can be downloaded or via the use of the command line:

```
# Download and untar directory
wget -qO- https://baileylab.brown.edu/MIPTools/download/test-data.tar.gz | tar -xvz
```

The test data set contains 5 directories that contain the test data, species resources, as well as project resources:

```
tree -FL 1 test-data

#> test-data
#> ├── DR1_project_resources/
#> ├── hg38_host/
#> ├── pf_species_resources/
#> ├── test_data/
#> └── test_design_resources_pf/
```

Test designs will be carried out on a few *Plasmodium falciparum* targets. The species resources directory includes resources for the *P. falciparum* genome and the host resources directory contain the resources for the human genome, which will be used as the host species.

Attention: The MIP design manual can be found here, for the time being, and an example walkthrough here. We plan on merging those documents here in the future…

CHAPTER

FOUR

ANALYSIS PIPELINE

This guide provides a walkthrough of some of the key analysis steps in analyzing molecular inversion probe data. We use a test data set composed of FASTQ files and a sample list.

Note: This guide does not cover the *download* or *demux* apps. The test data provided has already been demultiplexed.

4.1 Download Test Data

The test data set can be downloaded or via the use of the command line:

```
# Download and untar directory
wget -qO- https://baileylab.brown.edu/MIPTools/download/test-data.tar.gz | tar -xvz
```

The test data set contains 5 directories that contain the test data, species resources, as well as project resources:

```
tree -FL 1 test-data

#> test-data
#> ├── DR1_project_resources/
#> ├── hg38_host/
#> ├── pf_species_resources/
#> ├── test_data/
#> └── test_design_resources_pf/
```

4.2 Wrangle Data

After downloading the test data, the next step is to run the *wrangler app*. We first create a directory that will store our wrangler analysis and copy our sample list into said directory:

```
mkdir test-data/wrangler
cp test-data/test_data/sample_list.tsv test-data/wrangler/
```

We additionally define several parameters needed to wrangle data:

```
experiment_id='test_run'
sample_list='sample_list.tsv'
probe_sets_used='DR1,VAR4'
```

(continues on next page)

(continued from previous page)

```
sample_sets_used='JJJ'
cpu_number=10
min_capture_length=30
```

Next, we can run the *wrangler app*. For additional instructions on what each flag represents, consult the *man page* for the app or the built in documentation with `singularity run --app wrangler miptools_dev.sif -h`.

```
singularity run \
  -B test-data/DR1_project_resources:/opt/project_resources \
  -B test-data/test_data/fastq:/opt/data \
  -B test-data/wrangler:/opt/analysis \
  --app wrangler miptools_dev.sif \
  -e ${experiment_id} -l ${sample_list} -p ${probe_sets_used} \
  -s ${sample_sets_used} -c ${cpu_number} -m ${min_capture_length}
```

The *wrangler app* will save the main outputs as compressed files in the `wrangler` directory. There will additionally be a `nohup` file that contains errors and warning messages logged by the *wrangler app*. This file should be empty if the all went well. In our example run, the `nohup` file was empty and the main outputs were aggregated into the three files:

- `run_test_run_wrangled_20220314.txt.gz`
- `extractInfoByTarget.txt.gz`
- `extractInfoSummary.txt.gz`

Tip: After confirming the *wrangler app* successfully ran, we recommend you delete the `wrangler/analysis` directory. This will remove many small files and save space in the future.

```
rm -rf test-data/wrangler/analysis
```

4.3 Variant Calling

To further process our data and call and analyze variants, we will leverage an interactive Jupyter notebook by calling the *jupyter app*. Our main variant calling method uses the Freebayes software, a Bayesian genetic variant detector. While we have optimized the algorithm for calling on molecular inversion probes (MIPs), we use an interactive environment for calling and initial assessment to provide the user with greater customizability.

Before running the *jupyter app*, we must define a new directory in which we will run our variant calling pipeline:

```
mkdir test-data/variant
```

Then we can start our Jupyter notebook:

```
singularity run \
  -B test-data/DR1_project_resources:/opt/project_resources \
  -B test-data/pf_species_resources:/opt/species_resources \
  -B test-data/wrangler:/opt/data \
  -B test-data/variant:/opt/analysis \
  --app jupyter miptools_dev.sif
```

A series of instructions will be printed to the terminal on how to access the notebook. Follow these instructions to run the Jupyter notebooks in a web browser. For more information refer to the *FAQ of the jupyter app*. Next, navigate to the

`analysis` directory. The `analysis-of-test-data-Freebayes` notebook contains a demonstration of processing data, variant calling, and additional data analysis.

CHAPTER

FIVE

HPCC USE

In order to improve the computational efficiency of MIPTools, it is possible to use MIPTools via a high-performance computing cluster (HPCC). While we leave the details of HPCC use to the reader, briefly, clusters use a management and job scheduling system to organize user requests. Users may submit jobs to these management systems, which will in turn schedule and execute submitted jobs.

Note: This guide covers HPCCs configured to use slurm cluster manager. For HPCCs configured with another job manager, please review the documentation for the job manager of interest.

To submit a job via slurm, users can run the following:

```
sbatch <jobscript>
```

Each job script is a batch script with commands for the HPCC to execute. These files also contain configuration commands used by slurm. The bash shebang and these configuration commands make up what we call the header section of the script. An example is shown below:

```
#!/bin/bash

# Request a specific partition:
# The partitions available are: batch, gpu, and bigmem
#SBATCH --partition=batch

# Configure runtime, memory usage, and the number of CPUs:
#SBATCH --time=48:00:00
#SBATCH --mem=200G
#SBATCH --cpus-per-task=32

# Notify the user details about the job:
#SBATCH --mail-user=example@mail.com
#SBATCH --mail-type=ALL
```

The next section of the job script contains the bash code to be executed by the HPCC. This can contain a command to run a MIPTools app or some other MITPools command. For example, users could run the *wrangler app*:

```
# Paths to bind to container
project_resources=heome
fastq_dir=fastq
wrangler_dir=wrangler

# Wrangler options
```

(continues on next page)

(continued from previous page)

```
probe_sets_used='HeOME96'
sample_sets_used='JJJ'
experiment_id='example_id'
sample_list='sample_list.tsv'
min_capture_length=30

singularity run \
  -B ${project_resources}:/opt/project_resources \
  -B ${fastq_dir}:/opt/data \
  -B ${wrangler_dir}:/opt/analysis \
  --app wrangler miptools.sif \
  -p ${probe_sets_used} -s ${sample_sets_used} -e ${experiment_id} \
  -l ${sample_list} -c ${SLURM_CPUS_PER_TASK} -m ${min_capture_length}
```

This is just one of the many possible job scripts a user could use. Almost every aspect of the MIPTools pipeline could be configured to run via a HPCC. For more examples of job scripts, see the slurm-scripts repository. The MIPTools folder has scripts to run different MIPTools commands. Please feel free to add more scripts via a pull request!

CHAPTER

SIX

DOWNLOAD

6.1 Synopsis

```
singularity run [run options...] --app download <container> [app options...]
```

6.2 Description

Download data from the Illumina BaseSpace Sequence Hub.

6.3 Options

```
# Required
-i    The run ID of the data to download.

# Optional
-o    The path to the output directory.
-c    The path to the authentication credentials file.
-h    Print the help page.
```

6.3.1 Defaults

```
-o    Default: '/opt/analysis'
-c    Default: '/opt/resources/basespace.cfg'
```

6.3.2 Authentication Credential File

Note: Users must first authenticate their account in order to download data from the BaseSpace Sequence Hub.

The authentication credential file can be generated via the BaseSpace CLI. The `bs auth` command generates a configuration file that indicates an API server to contact and an access token to authenticate against BaseSpace Sequence Hub. The default API server defaults to Virginia, USA. Other options include Europe and the UK, among others. To override the default API server, use the `--api-server` option.

```
singularity exec [options] <container> bs auth --force

# Specify UK API server
singularity exec [options] <container> bs auth --force --api-server https://api.euw2.sh.
↪basespace.illumina.com
```

These commands will generate a configuration file: `basespace.cfg` in `${HOME}/.basespace`. In order for the download app to access the configuration file, you must copy the file into `base_resources`.

```
cp ${HOME}/.basespace/basespace.cfg base_resources
```

6.4 Examples

```
singularity run \
  -B base_resources:/opt/resources \
  -B downloaded:/opt/analysis \
  --app download miptools.sif -i 12345
```

CHAPTER

SEVEN

DOWNLOAD_SUPERSEDED

7.1 Synopsis

```
singularity run [run options...] --app download_superseded <container> [app options...]
```

7.2 Description

Download data from the Illumina BaseSpace Sequence Hub.

Warning: This app has been superseded by the *download app*, which uses the BaseSpace CLI for downloading data.

7.3 Options

```
# Required
-r    The run ID of the data to download.

# Optional
-h    Print the help page.
```

7.3.1 Download Destination

Data will be downloaded to `/opt/analysis`. A directory may be mounted to this path to customize the download destination.

7.3.2 Authentication Credential File

Note: Users must first authenticate their account in order to download data from the BaseSpace Sequence Hub.

An authentication token must be generated in order to download data from the BaseSpace Sequence Hub. In order to do so, you consult the *authentication credential file section of the download app*. Once generated authentication token must be copied to `base_resources/access-token.txt`.

Note: The *authentication credential file section of the download app* will generate a configuration file that indicates an API server to contact and an access token to authenticate against BaseSpace Sequence Hub. Only the **access token value** must be copied to `base_resources/access-token.txt`.

7.4 Examples

```
singularity run \
  -B base_resources:/opt/resources \
  -B downloaded:/opt/analysis \
  --app download_superseded miptools.sif -r 12345
```

CHAPTER

EIGHT

DEMUX

8.1 Synopsis

```
singularity run [run options...] --app demux <container> [app options...]
```

8.2 Description

Demultiplex data. Generate per-sample FASTQ files from the raw sequence data consisting of BCL files.

Warning: The demultiplexing software `bcl2fastq` is not shipped with prebuilt versions of MIPTools. To demultiplex BCL files, you must download the software, place it in the `programs` directory, and build MIPtools from *source*.

8.3 Options

```
# Required
-s    Path to the sample sheet for demultiplexing.

# Optional
-h    Print the help page.
```

8.3.1 Sample Sheet

The sample sheet must be present in the directory mounted to `/opt/analysis`.

8.4 Examples

```
singularity run \
  -B base_resources:/opt/resources \
  -B bcl_dir:/opt/data \
  -B fastq_root_dir:/opt/analysis \
  --app demux miptools.sif -s SampleSheet.csv
```

CHAPTER

NINE

DEMUX_QC

9.1 Synopsis

```
singularity run [run options...] --app demux_qc <container> [app options...]
```

9.2 Description

Compute quality control checks on demultiplexed data. Prints the total number of sequencing reads and how many were of undetermined indices, meaning that they contained indices that were not present in the sample file. Additionally prints how many of the undetermined index reads belong to possible primer pairs, i.e., they are likely due to errors in the provided sample list and not just faulty reads from the sequencer.

9.3 Options

```
# Required
-p    The sequencing platform used. Can either be 'nextseq' or 'miseq'.

# Optional
-h    Print the help page.
```

9.4 Examples

```
singularity run \
  -B base_resources:/opt/resources \
  -B downloaded:/opt/analysis \
  --app demux_qc miptools.sif -p 'nextseq'

singularity run \
  -B base_resources:/opt/resources \
  -B downloaded:/opt/analysis \
  --app demux_qc miptools.sif -p 'miseq'
```

CHAPTER

TEN

WRANGLER

10.1 Synopsis

```
singularity run [run options...] --app wrangler <container> [app options...]
```

10.2 Description

Run MIPWrangler on demultiplexed data.

10.3 Options

```
# Required
-e    A unique ID given to each sequencing run by the user.
-l    File providing a list of samples with associated information.
-p    Probe sets to be processed.
-s    Sample sets to be processed.
-x    Additional arguments passed to MIPWrangler mipSetupAndExtractByArm,
      which extracts sequences and stitches paired end reads to single
      sequences.

# Optional
-c    Number of available processors to use.
-f    Population fraction cutoff used by MIPWrangler.
-h    Print the help page.
-k    Keep intermediate files generated by MIPWrangler.
-m    Minimum capture length for stitching excluding probe arms.
-n    Starting number for MIP server.
-o    Absolute path to MIPWrangler run script.
-t    The threshold at which UMIs will be downsampled. For any MIPs with more
      UMIs than this threshold, the number of UMIs will be reduced to the
      threshold.
-w    Whether to apply a weight when randomly sampling UMIs. UMIs are
      weighed by their read counts.
```

10.3.1 Defaults

```
# Required
-x    Default: 'none'

# Optional
-c    Default: 1
-f    Default: 0.005
-k    Default: false
-m    Default: 100
-n    Default: 1
-o    Default: '/opt/bin/runMIPWranglerCurrent.sh'
-t    Default: 2000
-w    Default: false
```

10.4 Examples

```
# Define variables
probe_sets='DR1,VAR4'
sample_sets='JJJ'
stitch_options='--stitchGapExtend=1,--overWriteDirs'

# Run app
singularity run \
  -B project_resources:/opt/project_resources \
  -B fastqs:/opt/data \
  -B wrangler_dir:/opt/analysis \
  --app wrangler miptools.sif \
  -e 'example' -l 'sample_list.tsv' -p ${probe_sets} \
  -s ${sample_sets} -x ${stitch_options}

singularity run \
  -B project_resources:/opt/project_resources \
  -B fastqs:/opt/data \
  -B wrangler_dir:/opt/analysis \
  --app wrangler miptools.sif \
  -c 10 -e 'example2' -l 'sample_list.tsv' -p ${probe_sets} \
  -s ${sample_sets} -x ${stitch_options} -k
```

CHAPTER

ELEVEN

JUPYTER

11.1 Synopsis

```
singularity run [run options...] --app jupyter <container> [app options...]
```

11.2 Description

Open an interactive Jupyter Notebook. The notebook can be used for post-wrangler mapping and variant calling.

11.3 Options

```
# Optional
-d    The notebook directory.
-h    Print the help page.
-p    The port to be used to load the Jupyter Notebook.
```

11.4 Examples

```
singularity run \
  -B base_resources:/opt/resources \
  -B project_resources:/opt/project_resources \
  -B pf_species_resources:/opt/species_resources \
  -B wrangler_dir:/opt/data \
  -B variant_dir:/opt/analysis \
  --app jupyter miptools.sif
```

11.5 Accessing the Notebooks

When running this app, a series of instructions will be pasted to the terminal on how to access the notebook. Accessing the server will depend on whether the app was run via a remote server or via your local machine. In the case where the app was run via a remote server, you must forward the port to your local machine. At this point you may open a browser, paste the URL, and navigate to the `analysis` directory to access the notebooks.

11.6 FAQ

I can't connect to the remote server via ssh.
Make sure that you can access the server you are trying to connect to. In some cases, you may need to use a VPN.

How can I use this app on a HPCC?
If you are using a cluster computing system, the login node will likely be different from the compute node. In that case, the printed remote server will refer to the compute node and you will need to change this to the login node. For example, you may login in to the HPCC using: `ssh oa@login.hpcc.edu` and submit a job to the compute node `compute1`. In this case, the app may print `ssh -N -f -L localhost:9913:128.148.254.107:9913 oa@compute1.hpcc.edu`. You would change this to `ssh -N -f -L localhost:9913:128.148.254.107:9913 oa@login.hpcc.edu`.

CHAPTER

TWELVE

CHANGELOG

12.1 MIPTools (development version)

12.1.1 New Features

- When running the `wrangler` app, if the number of UMIs detected for a MIP is above a certain threshold, we reduce the UMI count to a lower value. This is done in order to increase the speed of our pipeline. Above a certain UMI count, the information becomes redundant (@arisp99, #40).
- Add an additional argument to the `wrangler` app to control the population clustering fraction cutoff defined by MIPWrangler (@arisp99, #39).
- Add the capability to freeze software version numbers when building the container. Additionally, the version number for key software tools has been fixed (@arisp99, #32).
- Install mipscripts, which contains additional tools for analysis pipelines.
- Perform additional argument parsing to ensure arguments are formatted correctly (#28, #37).
- New `download` app supersedes the previous `download` app, which has been renamed to `download_superseded`. The new app improves the method for downloading data from the Illumina BaseSpace Sequence Hub by using the official command line tool (@arisp99, #25, #13).

12.1.2 Bug Fixes

- Upgrade C and C++ compiler versions (#43).
- Don't install conda and mamba packages using defaults as this can cause the install process to hang.
- Upgrade `libgfortran4` to `libgfortran5` (#38).
- Let Freebayes run with only one CPU thread (#33).
- Fix error when app arguments have whitespace characters (#26, #37).
- Fix missing file error when MIP arms file is created from the MIP info dictionary (@aydemiro, #23).
- Improve sample sheet preparation. Avoid errors when sample file columns are empty. Throw an error if there are invalid samples or input fields (@aydemiro, #22).
- Fix build failure due to dependency changes in the McCOILR R package (#7).

12.1.3 Maintenance

- Remove the `msa2vcf` program and other conversion tools (#35).
- Reduce size of image by deleting source code after installation of programs.
- Remove sequence aligners (#35).
- Remove unused analysis settings files (#35).
- Install programs from GitHub instead of storing source code (@arisp99, #36).
- Update LICENSE year.
- Store containers using an HTTP directory (#12).
- Remove duplicated files.
- Improve bash errors.
- Make strings human readable (@arisp99, #5).

12.1.4 Documentation Overhaul

- Add guides on *probe design*, *data analysis*, and *HPCC use*.
- Generate online documentation using Sphinx and Github Pages.
- Improve app documentation.
- Add doc-strings to python functions.
- Improve clarity of README and add additional instructions on downloading or *building the container*.

12.2 MIPTools 0.4.0

- Latest stable build.

CHAPTER

THIRTEEN

LICENSE

MIPTools is licensed under the MIT License.

A short and simple permissive license with conditions only requiring preservation of copyright and license notices. Licensed works, modifications, and larger works may be distributed under different terms and without source code.

Permissions

- Commercial use – The licensed material and derivatives may be used for commercial purposes.
- Modification – The licensed material may be modified.
- Distribution – The licensed material may be distributed.
- Private use – The licensed material may be used and modified in private.

Conditions

- License and copyright notice – A copy of the license and copyright notice must be included with the licensed material.

Limitations

- Liability – This license includes a limitation of liability.
- Warranty – This license explicitly states that it does NOT provide any warranty.

See more information on choosealicense.com ⇒

```
MIT License

Copyright (c) 2023 Bailey Lab

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
```

(continues on next page)

(continued from previous page)

```
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

CHAPTER

FOURTEEN

NEED HELP?

- Join our
- See our

INDEX

M