mappa™ Analysis Pipeline User Guide (for Version 1.0)

I. Introduction

mappa Analysis Pipeline is bioinformatic software for analyzing single-cell RNA-seq NGS data generated using the ICELL8® cx Single-Cell system or the ICELL8 Single-Cell system on the single-cell full-length transcriptome or single-cell differential expression (3′ DE or 5′ DE) workflows.

The program takes input data from sequencing and the well-list file generated by any CellSelect® software and outputs an HTML report, with results typical to single-cell analysis, plus other files, such as a gene matrix, to continue further analysis. R data objects with pre-computed results based on recommended parameters are also output. Either the standard output files or the R data objects can serve as input for hanta™ R kit (hanta), another bioinformatic software package provided by Takara Bio.

mappa software is written in Python and can be run either via a GUI or command-line interface.

II. Before You Begin

A. Supported operating systems

mappa software is designed to be installed on a server running Linux. The following versions of Linux have been tested and are supported for use with mappa software:

- CentOS 6.9 & 6.10
- RedHat 7.5
- Ubuntu 17

B. Hardware requirements

For analyzing the output of Illumina® NextSeq® High-Output sequencing data analysis, the following server requirements (or better) are recommended:

- CPU: 24-cores
- RAM: 64 GB
- Free hard drive space: 500 GB

Testing was also done on MiniSeq™, MiSeq®, HiSeq®, and NovaSeq™ datasets.

- MiniSeq or MiSeq—less computational power may be needed than the specifications described above for NextSeq output
- HiSeq or NovaSeq—requires more computational power than described above for NextSeq output.

Precise hardware requirements were not determined for output from these datasets. Support for performance issues of the servers in conjunction with these dataset types may be limited.

C. User account requirements

mappa software can be installed in two different access scenarios; the user account requirement depends on which scenario you wish to implement in your environment.

- For use by a single user (single Linux username/account), mappa software can be installed and run by any account type with install and executable privileges (regular or root access).
- For use by multiple users (accounts), mappa software can either be installed singly across the multiple accounts (regular or root access) or installed for system-wide access by an administrator with root access.

D. Additional hardware and software dependencies and recommendations

- **ICELL8 cx Single-Cell System or ICELL8 Single-Cell System**
- **Bash UNIX shell**
- **Internet connectivity on the server**

  The installation process requires Internet connectivity as it sources scripts from GitHub, Bioconda, and CRAN and downloads genome information from Ensembl. Please ensure that internet connectivity is available on the UNIX server while installing.

- **Conda**

  mappa software leverages the open-source package manager Conda for installation of the pipeline and its dependencies. Any tools and applications required by the pipeline are installed through Conda inside a local environment created specifically for mappa software.

  If Conda is not currently installed on the server, instructions to do so can be found at https://conda.io/miniconda.html. We recommend installation of the lightweight version for Python 3.7+ (typically the 64-bit bash installer), also called Miniconda3 (version 4.5.0 or later).

- **Keyboard, monitor, and mouse directly into the server, or a remote access program**

  mappa software must be run on the Linux server. If users do not have direct console access, a remote access program that enables a Virtual Network Computing (VNC) connection is required through a program such as RealVNC (www.realvnc.com), TightVNC (www.tightvnc.com), Tiger VNC (tigervnc.org), or similar.

  For more information on VNC, along with other VNC clients that can be used, please see the Wikipedia entry at https://en.wikipedia.org/wiki/Virtual_Network_Computing.

E. Required Input File

- Paired-end read FASTQ files
- CellSelect well-list file—a well selection text file derived from the CellSelect software which contains well-level sample information. A typical CellSelect well-list file should have no more than 1,728 wells.

III. Software overview

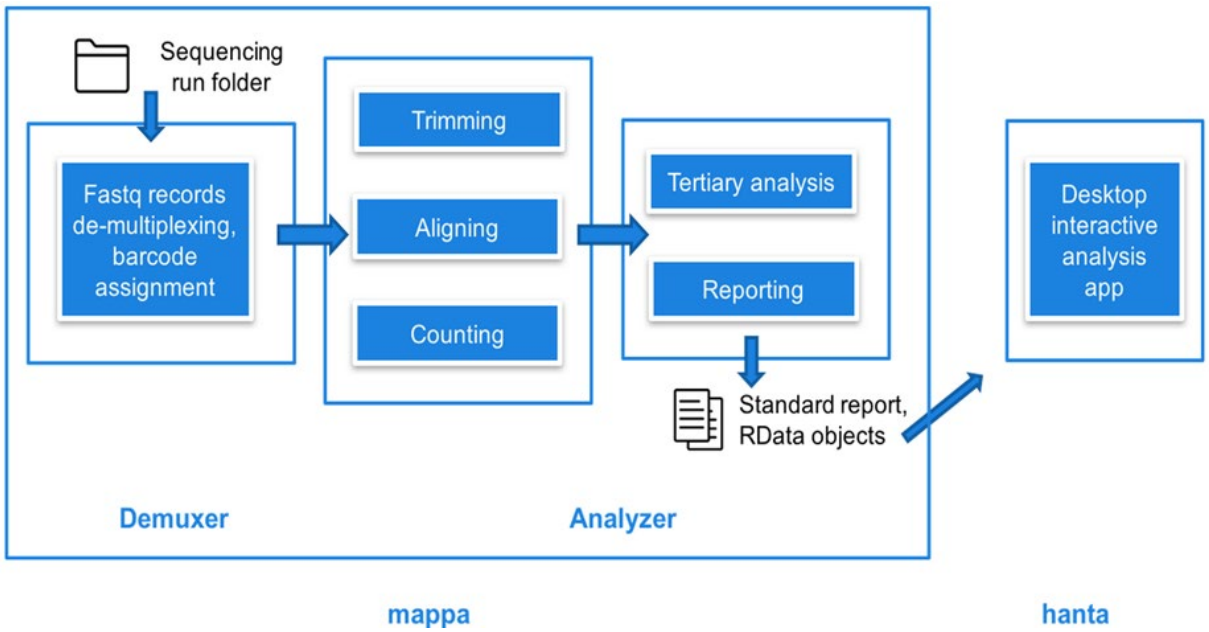


Figure 1. High-level analysis workflow.

Figure 1 depicts the high-level workflow of the analysis provided by mappa software and how its output can be carried over to the interactive R-kit, hanta.

mappa software consists of two main parts, the demultiplexer (demuxer) and the analyzer.

- The demuxer extracts the barcode from the sequencing data (based on the protocol) and writes it into FASTQ files at the end of the read name. There are two options: the default which leaves the barcode-assigned reads in combined FASTQ files (which saves time during subsequent analysis), or an option to split the data up into barcode-level FASTQ files which can be carried over to other analysis pipelines.
- The demuxer then sends the demultiplexed data to the analyzer, which performs the following functions:
  - Read trimming (using Cutadapt)
  - Genome alignment (using the STAR aligner)
  - Counting (using Subread)
  - Summarization (using custom scripts)
  - Generating an HTML report (using hanta)

The demuxer and analyzer can be invoked using a graphical user interface (GUI), called the mappa launcher, or can be run on the command line.

IV. Installation & configuration options

The mappa software is available for download on the ICELL8 software portal at takarabio.com/ICELL8-software. An email will be sent to you with information on downloading the install script and a unique password required to run the installation on your server.

A. Verifying Conda installation

The following steps can be used to verify that Conda is installed properly on the server.

1. Type the following command in at the prompt in any directory location on the server.

   ```
   conda -V
   ```

   If conda is successfully installed, it should return text with the version number.

   e.g.,

   ```
   conda 4.5.11
   ```

2. Check to see if the base conda environment can be activated.

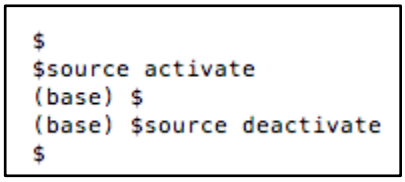
   

   Figure 2. High-level Screenshot of the Linux command-line showing a successful check of the base conda environment.

   a. Type the following command into the command-line Linux prompt on the server:

      ```
      source activate
      ```

      Or, if that returns an error, try:

      ```
      conda activate
      ```

      A successful conda install will result in a change in the prompt, as shown in Figure 2 (above). If the conda installation was not completed as required, both commands will return error messages.

   b. If conda is successfully installed and the prompt changed as displayed in Figure 2, type the following command to return to the default Linux prompt:

      ```
      source deactivate
      ```

      or

      ```
      conda deactivate
      ```

      This command will take the user back to the Linux prompt and out of the conda environment.

3. Installation of Miniconda3 typically adds the location of its installation to the user's system environment. This is also required for the successful installation of mappa software.

   The following steps can be used to verify that the conda `$PATH` is configured correctly.

   a. Open the file `.bash_profile`, which will be located in the home directory (for an individual user account):

```
more ~/.bash_profile
```

b. Verify a line similar to the following is showing in the file:

```
export PATH="/home/<USERNAME>/miniconda3/bin:$PATH"
```

where <USERNAME> is replaced by the username of the account that installed conda.

e.g., username is 'myacct':

```
export PATH="/home/myacct/miniconda3/bin:$PATH"
```

If the line isn't displaying or no .bash_profile file exists, it will need to be manually created and populated. See Section IV.C for more information on how to manually create and populate a .bash_profile file.

B. Installation

1. Download the installation script (takarabio_Linux64_installer.sh), following the directions on the thank you page or the email sent after submitting the sign-up form on the mappa Analyzer Pipeline product page.
2. Move or copy the installation script onto the Linux server into the directory location where you want to install the mappa software.

> **NOTE**: The account logged into while doing the installation must have read/write privileges to the install directory chosen.

3. From the same directory location in step #2, run the following command:

```
bash takarabio_installer.sh mappa <PASSWORD>
```

<PASSWORD> will be replaced by the unique password included in the sign-up confirmation email.

> **NOTE**: The installation process at this point will take anywhere from 90 minutes to 3 hours to complete, depending on the computational capacity of the server and the download speed of the internet connection.
>
> No further user input is required until the install is complete. The installation procedure may be left to run overnight, with the user not present at the console. If desired, the user can work in another terminal window simultaneous to the install.

Once the installation is complete, a message will display saying mappa software is ready for use.

```
The pipeline, dependencies and human genome were successfully installed
The pipeline is now ready to use
```

Figure 3. Text to console illustrating a successful mappa software install on the Linux server.

There should be a folder named `mappa` in the directory into which the software was installed (Section IV.B.2) which contains files and directories required by the pipeline's scripts.

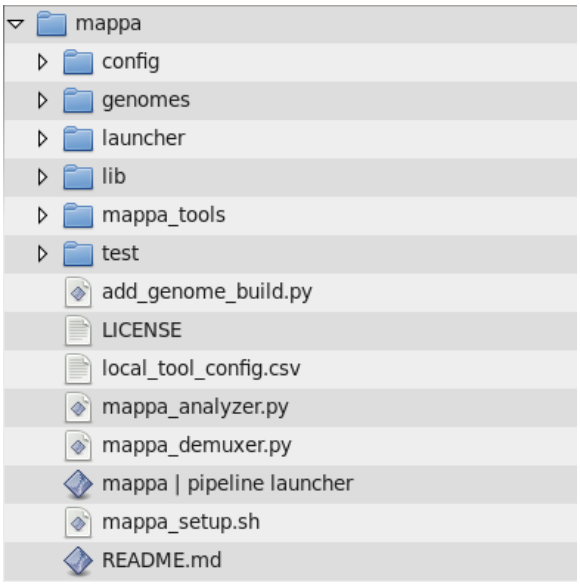


Figure 4. The sub-directory and files list of the mappa\ folder post-install.

C. (Optional) Set up a `$MAPPA_HOME` environmental variable

For ease of use, it's recommended that the mappa software install directory location be added to the `.profile` file or `.bash_profile` as a permanent environmental variable.

> **NOTE**: A `~/.profile` or `~/.bash_profile` file may already exist in the home directory of the install account. If either exist, modify the existing file. If neither exist, then create `.bash_profile`.

Example:

If your account name is 'myacct', the absolute pathname for myacct's home directory is `/home/myacct`, and mappa software was installed in the `~/bin` directory, edit `.bash_profile` to add the following line:

```
export MAPPA_HOME=/home/myacct/bin/mappa
```

Once added to the profile, you will either need to log out and back in to the account, or load the file in with one of the following commands (depending on which file was modified):

```
# Refresh the profile
source ~/. profile
# or
source ~/.bash_profile
```

The phrase `$MAPPA_HOME` can then be used as an alias shortcut to reference `/home/myacct/bin/mappa`.

Example:

Running the following command logged in as 'myacct':

```
cd $MAPPA_HOME
```

will change directories to `~/bin/mappa`.

For more information on setting a custom environment variable, see a UNIX user manual or a forum post like https://stackoverflow.com/a/7502128.

NOTE: Subsequent references to `$MAPPA_HOME` in this document refer to the full path where mappa software is installed.

D. Updating the pipeline scripts

To update mappa, run the following two commands in sequence:

```
cd $MAPPA_HOME
```

```
sh mappa_setup.sh update
```

E. Add a genome build

The human genome (Ensembl hg38 release 94) is built as default with the pipeline, but the genomes of other species can be loaded into the software post-install.

To do so, download the following two files for the genome of interest:

1. The FASTA file containing all the sequences (chromosomes and contigs)
2. The GTF file containing the annotation, importantly the gene information for analysis
3. Run the script:

```
$MAPPA_HOME/add_genome_build.py -g <common_species_name> -f
<FASTA_FILENAME> -a <GTF_FILENAME>
```

Where `<common_species_name>` is the name of the genome being added (e.g., fruitfly) and the `<FASTA_FILENAME>` and `<GTF_FILENAME>` are the exact file names and path(s) for the FASTA and GTF files, respectively, on the server.

For additional help with this script, type:

```
$MAPPA_HOME/add_genome_build.py -h
```

Example, using the fruit fly genome from Ensembl.org (https://uswest.ensembl.org/info/data/ftp/index.html)

1. Download the FASTA file (`Drosophila_melanogaster.BDGP6.dna.chromosome.2L.fa`) using FTP:

   ```
   wget ftp://ftp.ensembl.org/pub/release-94/fasta/drosophila_melanogaster/dna/Drosophila_melanogaster.BDGP6.dna.toplevel.fa.gz
   ```

   or copy and paste the URI:

   ftp://ftp.ensembl.org/pub/release-94/fasta/drosophila_melanogaster/dna/Drosophila_melanogaster.BDGP6.dna.toplevel.fa.gz

   into the address bar of a web browser.

2. Download the GTF file (Drosophila_melanogaster.BDGP6.94.gtf):

   ```
   wget ftp://ftp.ensembl.org/pub/release-94/gtf/drosophila_melanogaster/Drosophila_melanogaster.BDGP6.94.gtf.gz
   ```

   or copy and paste:

   ftp://ftp.ensembl.org/pub/release-94/gtf/drosophila_melanogaster/Drosophila_melanogaster.BDGP6.94.gtf.gz

   into the address bar of a web browser.

3. If the downloaded files are stored in `~/ensembl` directory in "myacct", run the following script:

   ```
   $MAPPA_HOME/mappa_tools/bin/python3 $MAPPA_HOME/add_genome_build.py -g fruitfly -f ~/ensembl/Drosophila_melanogaster.BDGP6.dna.chromosome.2L.fa -a ~/ensembl/Drosophila_melanogaster.BDGP6.94.gtf
   ```

NOTE: As FASTA and GTF files are a standard file format, files from any source should work with this script. However, this script has only been tested on genomes downloaded from Ensembl.

If a problem is encountered using files from another source it is recommended to try the import using the Ensembl file versions of the genome.

F. Uninstall

mappa software can be uninstalled by deleting the `mappa/` directory defined as `$MAPPA_HOME` from the server.

If `$MAPPA_HOME` has been defined in `.profile` or `.bash_profile`, edit the file to remove the reference to `$MAPPA_HOME` as well.

V. Running the pipeline

The pipeline can be run in two ways:

A. Using a graphical user interface (GUI)

B. On the command-line

A. GUI

The mappa GUI is a program called launcher. Since it's graphical, it should be accessed by connecting to the Linux server using a VNC remote access tool (see Section II.D, above).

For additional information on the process of setting up the server and client for remote access, please see Appendix B.

1. Connect remotely or via a graphical interface on the server console to the Linux server where mappa software is installed.
2. Once connected to the server, navigate using the file browser functionality to the folder in which mappa software was installed. In Figure 5 (below), the mappa install folder is located on the user's desktop.

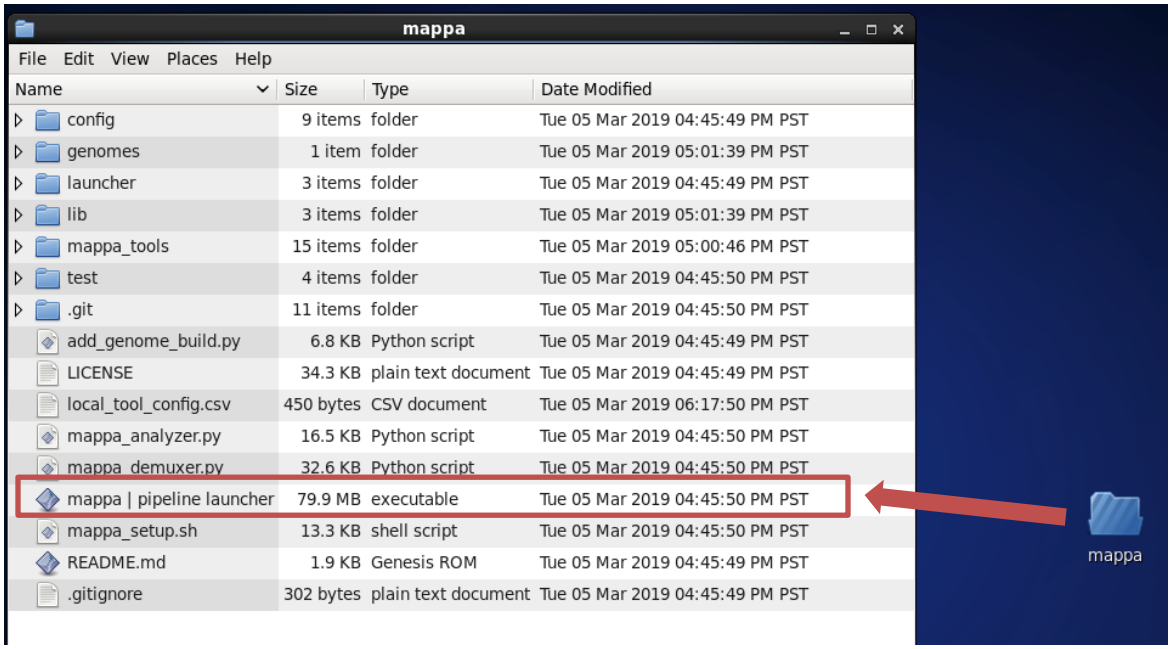


Figure 5. The `mappa\` folder on the desktop, opening it to highlight the `mappa | pipeline launcher` executable file.

3. In the install folder, locate `mappa | pipeline launcher` and double-click on it to run the UI launcher.
4. This will open the UI mappa pipeline launcher (Figure 6, below).

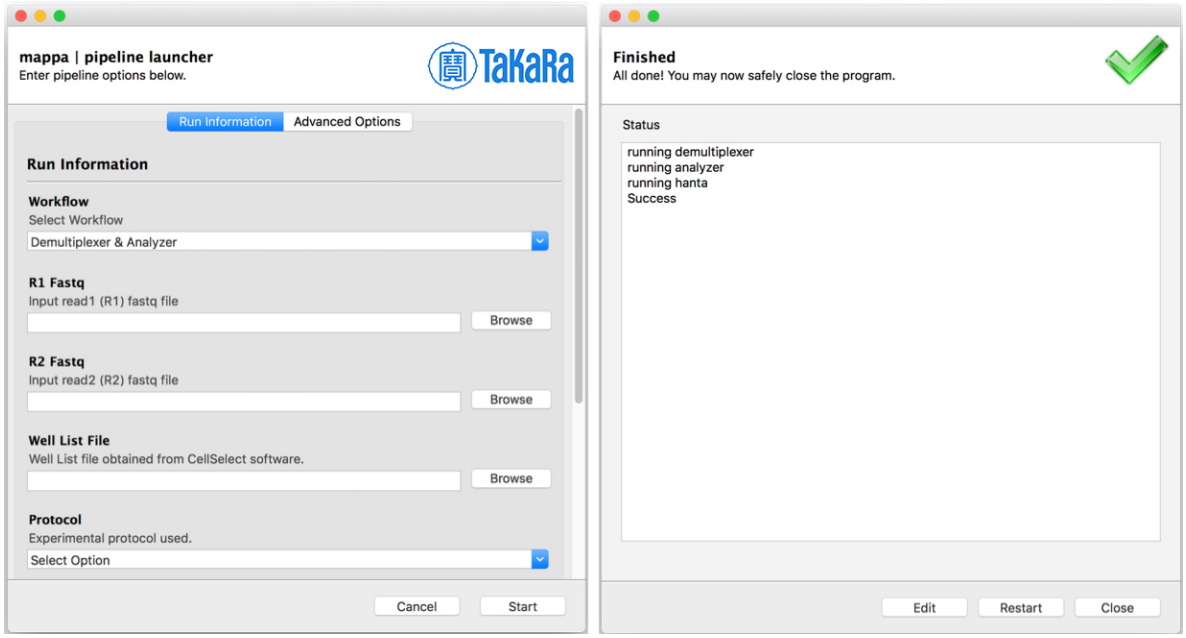


Figure 6. The mappa GUI.

5. The launcher has two tabs, *Run Information* and *Advanced Options*.
    a. The *Run Information* tab takes as input:
        - The workflow (default being 'Demultiplexer & Analyzer')
        - The input R1 & R2 FASTQs
        - The well-list file
        - The protocol: for example, full-length, 3' DE, or 5' DE
        - The genome to use for alignment (default: human hg38)
        - The directory location in which to output the results
        - A name for a new analysis folder
    b. The *Advanced Options* tab can be used to modify parameters such as number of cores available for analysis, number of mismatches to consider during barcode assignment, etc. This tab is intended for advanced users only.

    Once all the fields have been appropriately filled, clicking on [Start] launches the pipeline. The process typically takes a few hours to run and upon successful completion displays a corresponding message as shown in the right panel of Figure 6 (above).

B. Command line

> **NOTE**: See Section VI.B for an example of the full syntax for the command line scripts.

In general, you would first run the demuxer (`mappa_demuxer.py`) to obtain the demultiplexed FASTQ files.

These files are then used as input to run the analyzer (`mappa_analyzer.py`) to obtain the HTML report, stats file, genematrix and R data objects (described in detail in Section VII).

These scripts can be launched from any location (working directory) on the Linux server where mappa software is installed.

The full list of arguments can be accessed using the `-h` option (example in Figures 7 & 8, below.)

```
%MAPPA_HOME%/mappa_demuxer.py -h
```

```
$Desktop/mappa/mappa_tools/bin/python3 Desktop/mappa/mappa_demuxer.py -h
usage: mappa_demuxer.py [-h] -i R1_FILE -p R2_FILE -b BCS_FILE -t
                        {3pDE,3pDE_UMI,5pDE,Full_length} -o OUT_DIR
                        [-n N_THREAD] [-s] [-m {0,1}] [-w CHIP_BARCODES_FILE]
                        [-u UMI_LEN] [-k] [-r]

Script: mappa_demuxer.py Script to de-multiplex barcoded reads from sequence
data stored in fastq files. User options are designed to simplify de-
multiplexing for experiments derived from Takara protocols. Barcode (and
optionally, UMI) sequences are extracted and stored in the read name. Users
may specify whether the resulting de-multiplexed data are merged, or split
into individual barcode-level files.

required arguments:
  -i R1_FILE, --input_fastq R1_FILE
                        Input read1 (R1) fastq file (barcode)
  -p R2_FILE, --paired_fastq R2_FILE
                        Input read2 (R2) fastq file
  -b BCS_FILE, --barcodes_file BCS_FILE
                        WELL_LIST file obtained from CellSelect software
  -t {3pDE,3pDE_UMI,5pDE,Full_length}, --type_of_experiment {3pDE,3pDE_UMI,5pDE,Full_length}
                        Experimental protocol used
  -o OUT_DIR, --output_dir OUT_DIR
                        An output directory to be created to store results

optional arguments:
  -h, --help            show this help message and exit
  -n N_THREAD, --n_processes N_THREAD
                        Number of processes to spawn during execution
                        [Default: 1]
  -s, --split_fastqs    Using this argument will split the input fastqs to
                        barcode-level fastqs[Default: FALSE]
  -m {0,1}, --mismatch {0,1}
                        Number of mismatches to allow in the barcode [Default:
                        1]
  -w CHIP_BARCODES_FILE, --all_well_barcodes_file CHIP_BARCODES_FILE
                        Rare case where a custom chip/barcode set is used,
                        input a file with all the barcodes, to calculate
                        background
  -u UMI_LEN, --umi_length UMI_LEN
                        If protocol is 3pDE_UMI, specify UMI length [Default:
                        14, NA for other protocols]
  -k, --keep_temp       Using this argument will blockfile cleanup and keep
                        all temporary files [Default: FALSE]
  -r, --results_map     Using this argument will save a results map file. An
                        explicit map between read names and barcode/UMI
                        [Default: FALSE]
$
```

Figure 7. The output of `mappa_demuxer.py -h` at the command line.

```
%MAPPA_HOME%/mappa_analyzer.py -h
```

```
$Desktop/mappa/mappa_tools/bin/python3 Desktop/mappa/mappa_analyzer.py -h
usage: mappa_analyzer.py [-h] -i R1_FILE -g {hg38} -o OUT_DIR [-n N_THREAD]
                         [-p R2_FILE] [-d DEMUX_COUNTS]
                         [-c {exon,exon_intron}] [-k]

Script: mappa_analyzer.py Script to perform single-cell analysis on de-
multiplexed fastq files. The input to this script are files output by
mappa_demuxer script. The fastq files are expected to contain the barcode info
in the read name. The modules currently included are: - Trimming (cutadapt) -
Alignment (STAR) - Counting (featureCounts) - Summarization (TBUSA)

required arguments:
  -i R1_FILE, --input_fastq R1_FILE
                        De-muxed read1 (R1) fastq file
  -g {hg38}, --genome {hg38}
                        Choose the name of the genome to be used for analysis
  -o OUT_DIR, --output_dir OUT_DIR
                        An output directory to be created to store results

optional arguments:
  -h, --help            show this help message and exit
  -n N_THREAD, --n_processes N_THREAD
                       Number of processes to spawn during execution
                       [Default: 1]
  -p R2_FILE, --paired_fastq R2_FILE
                        De-muxed read2 (R2) fastq file
  -d DEMUX_COUNTS, --demux_counts DEMUX_COUNTS
                        Use counts file from demux script output
  -c {exon,exon_intron}, --counting_type {exon,exon_intron}
                        Summarize gene counts by alignments from exons-only
                        ("exon") or both exons and introns ("exon_intron",
                        recommended for Nuclei)
  -k, --keep_temp       Using this argument will stop file cleanup and keep
                        all temporary files [Default: FALSE]
$
```

Figure 8. The output of `mappa_analyzer.py -h` at the command line.

C. Processing time

The time taken by the pipeline will vary widely based on the specifications of the server on which it is run. During testing, a MiSeq run (~25M read pairs) typically took about 1–1.5 hrs, and a NextSeq High Ouput run (~400M read pairs) typically took ~20–25 hours to complete.

VI. Test dataset

The pipeline comes with a test dataset, stored under the main installation folder in a sub-folder called `$MAPPA_HOME/test/` (Figure 9, below). This dataset can be used to test the running of the pipeline end-to-end and will provide a sample of the output files.

NOTE: This is a small data set and should not be used for inference purposes. The statistics and plots will be meaningful only with a real dataset.

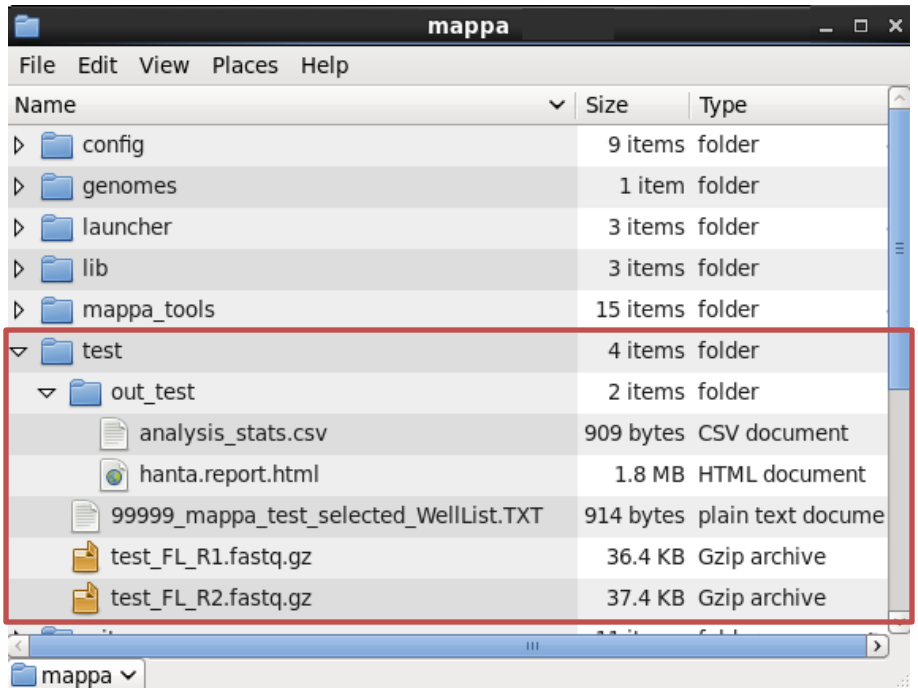


Figure 9. The `test/` folder under `$MAPPA_HOME`. The sample `*.fastq.gz` files and example output directory `out_test/` can be found there.

Example outputs (report and stats only) from the test data set are also stored under `$MAPPA_HOME/test/out_test/` for reference.

A run using the test data can be started using either the UI launcher (Figure 10) or the command line (below).

The test run using either method should take ~5–10 minutes to complete successfully. Once completed, the output folder (`install_test/` in the examples) will be located under the directory specified as 'Output Directory' (`/tmp/mappa_testing/`, Figure 10; `Desktop/` as part of the parameter `-o Desktop/install_test` in the demuxer command, Figure 11).

A. (Example) Test data through mappa launcher UI

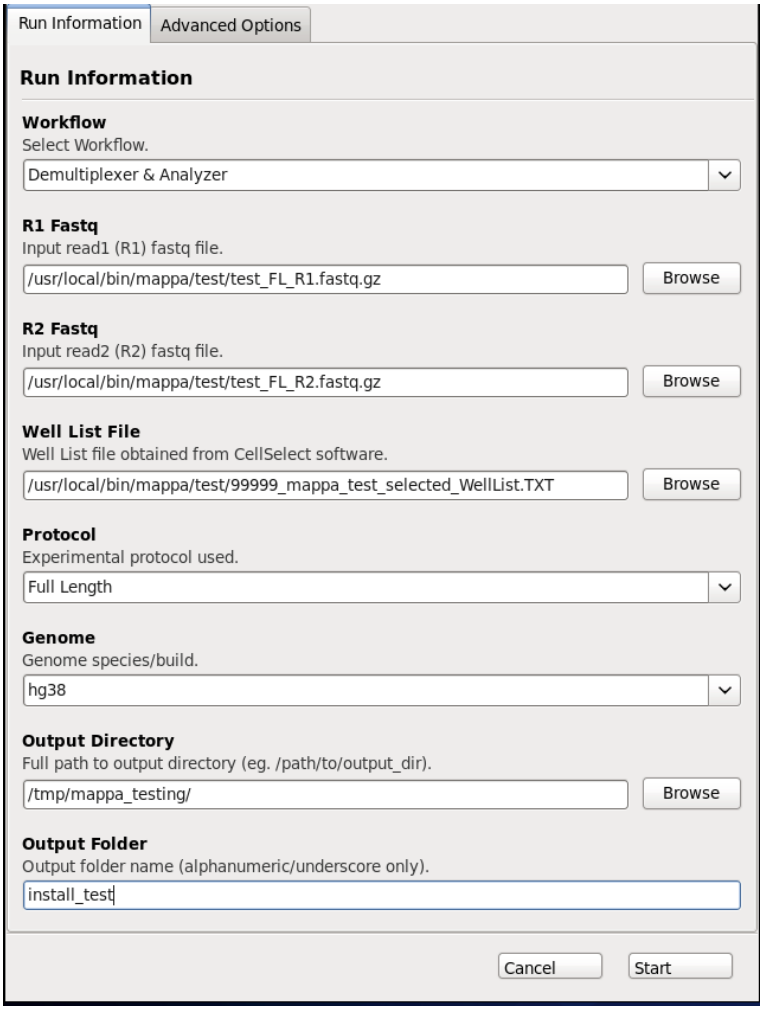


Figure 10. Configuration of the fields in the UI launcher for the test dataset files.

The parameters defined in the mappa launcher fields in Figure 10 are listed below. The paths are written as `$MAPPA_HOME` being `/usr/local/bin/`.

Table I. Parameter information for example mappa launcher run of the provided test dataset in Figure 10.

Field name	Input value
Workflow	`Demultiplexer & Analyzer`
R1 Fastq	`/usr/local/bin/mappa/test_test_FL_R1.fastq.gz`
R2 Fastq	`/usr/local/bin/mappa/test_test_FL_R2.fastq.gz`
Well List File	`/usr/local/bin/mappa/test/99999_mappa`
Protocol	`Full Length`
Genome	`hg38`
Output Directory	`/tmp/mappa_testing/`
Output Folder	`install_test`

B. (Example) Test data through the command line

1. Run the demultiplexer script.

```
$MAPPA_HOME/mappa_tools/bin/python3 $MAPPA_HOME/mappa_demuxer.py \
   -i $MAPPA_HOME/test/test_FL_R1.fastq.gz \
   -p $MAPPA_HOME/test/test_FL_R2.fastq.gz \
   -b $MAPPA_HOME/test/99999_mappa_test_selected_WellList.TXT \
   -o Desktop/install_test \
   -n 8
```

```
$Desktop/mappa/mappa_tools/bin/python3 Desktop/mappa/mappa_demuxer.py \
> -i Desktop/mappa/test/test_FL_R1.fastq.gz \
> -p Desktop/mappa/test/test_FL_R2.fastq.gz \
> -b Desktop/mappa/test/99999_mappa_test_selected_WellList.TXT \
> -t Full_length \
> -o Desktop/install_test \
> -n 8
```

Figure 11. Example syntax for running the demux script on the provided test data via the command line.

2. Once the demuxer is finished, run the data through the analysis script.

```
$MAPPA_HOME/mappa_tools/bin/python3 $MAPPA_HOME/mappa_analyzer.py \
   -i $MAPPA_HOME/install_test/install_test_demuxed_R1.fastq \
   -p $MAPPA_HOME/install_test/install_test_demuxed_R2.fastq \
   -g hg38 \
   -o Desktop/install_test/analysis \
   -n 8
   -d Desktop/install_test/install_test_counts_all.csv
```

```
$Desktop/mappa/mappa_tools/bin/python3 Desktop/mappa/mappa_analyzer.py \
> -i Desktop/install_test/install_test_demuxed_R1.fastq \
> -p Desktop/install_test/install_test_demuxed_R2.fastq \
> -g hg38 \
> -o Desktop/install_test/analysis \
> -n 8 \
> -d Desktop/install_test/install_test_counts_all.csv
```

Figure 12. Example syntax for running the analyzer script on the provided test data via the command line.

VII. mappa output

The pipeline produces a list of output files that serve two purposes:

1. Summarization of results using typical statistics and plots.
2. For facilitating further analyses using our interactive R kit, hanta, or any other tertiary analysis tool.

The output folder, `install_test/`, should include files and sub-folders similar to Figure 13, below.

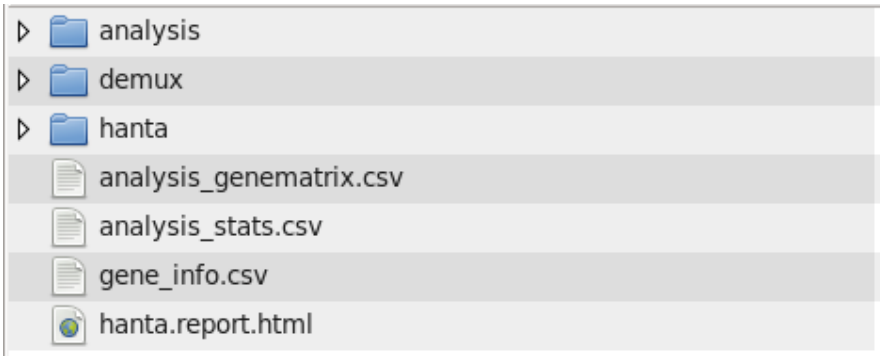
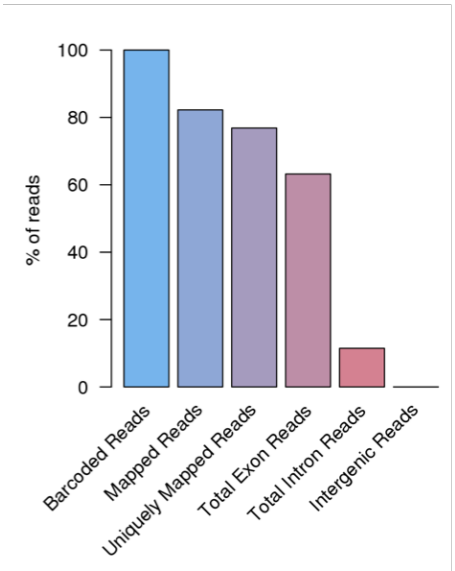


Figure 13. Folders and files of the output directory.

A. HTML report

The HTML report is generated by the report generator in hanta using standard parameters and contain the example statistics and plots listed below. For complete details, please see the hanta R kit User Guide.

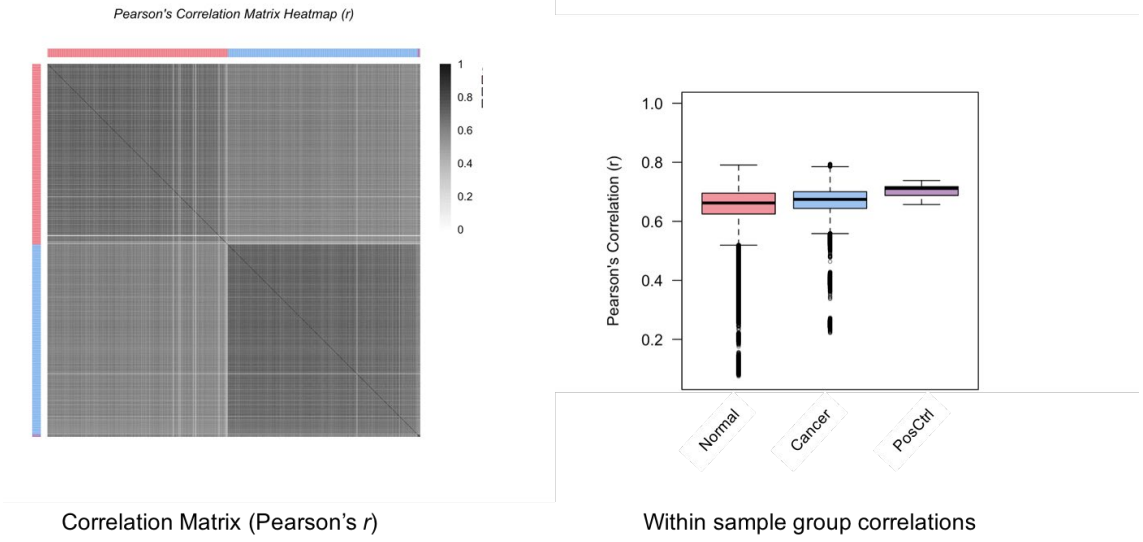
1. Experiment overview



	Read Count	% of Barcoded Reads
Barcoded Reads	31,968,709	100.00
Trimmed Reads	31,089,379	97.25
Mapped Reads	26,288,797	82.23
Uniquely Mapped Reads	24,570,861	76.86
Multimapped Reads	1,717,936	5.37
Total Exon Reads	20,209,407	63.22
Unique Exon Reads	19,134,255	59.85
Ambiguous Exon Reads	1,075,152	3.36
Total Intron Reads	3,670,550	11.48
Unique Intron Reads	3,286,235	10.28
Ambiguous Intron Reads	384,315	1.20
Intergenic Reads	0	0.00
Additional Information		
Mitochondrial Reads	1,111,817	3.48
Ribosomal Reads	480,833	1.50

Figure 14. Sample experiment overview section of the HTML report.

2. Correlation analyses



Correlation Matrix (Pearson's *r*) Within sample group correlations

Figure 15. Example correlation analyses section of the HTML report.

3. Various counts like reads, genes, Mito, Ribo.

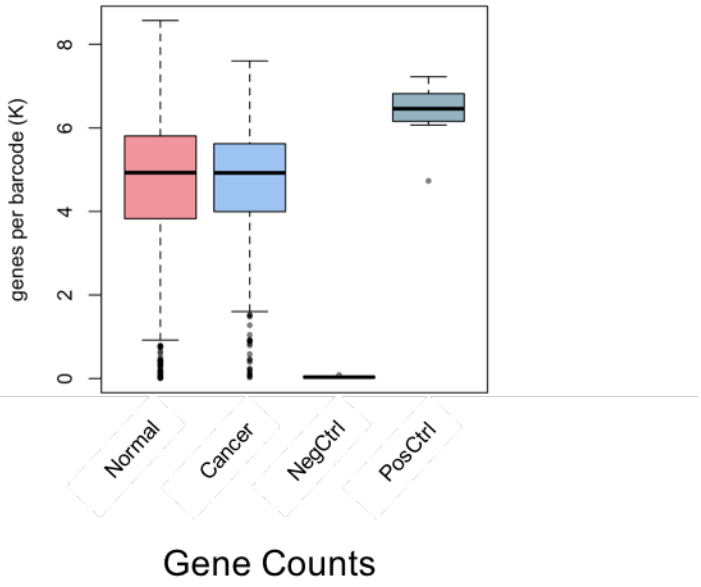


Figure 16. Example gene counts chart from the HTML report.

4. Clustering tables

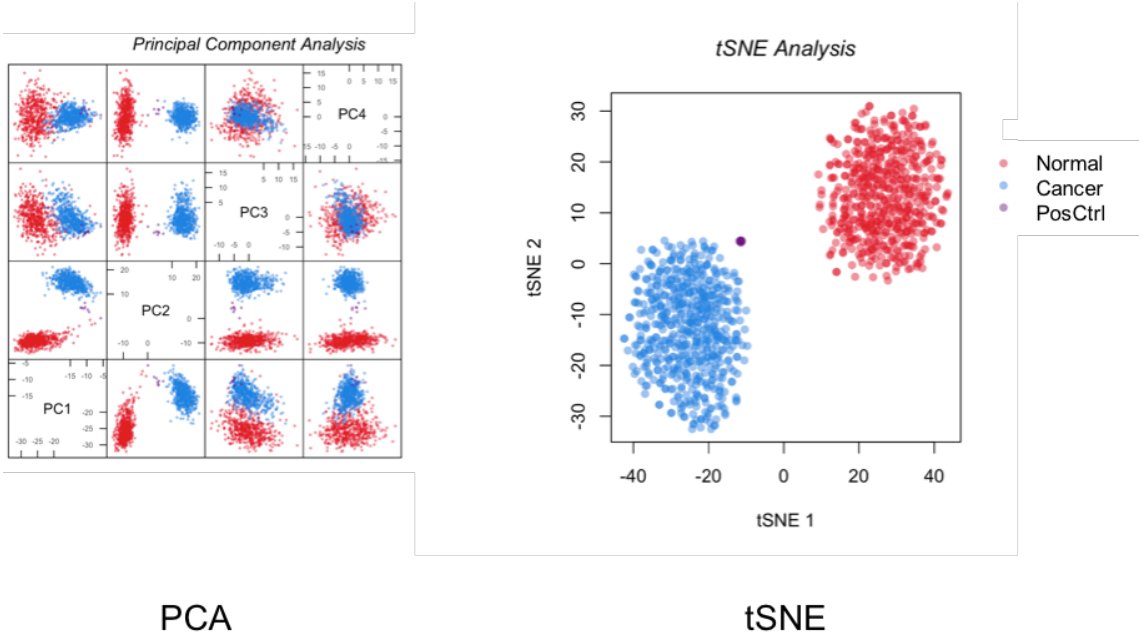


Figure 17. Example clustering tables from the HTML report.

B. Stats file

The stats file is provided in CSV format and contains barcode-level statistics across the analysis pipeline. Starting from barcoded reads, it summarizes the number of reads after every step in the pipeline: for example, trimmed reads, mapped reads, exon/intron/intergenic reads, mitochondrial reads, ribosomal reads, etc. It also lists the number of genes detected per barcode.

Barcode	Sample	Barcoded_Reads	Trimmed_Re	Unmapped_I	Mapped_Rea	Multimappe	Uniquely_Ma	Mitochondri
TTCGTAATCGTTGGTT	Sample	147367	146162	28892	117270	9740	107530	948
AGTACCATACCGAATT	Sample	293358	291002	33553	257449	17467	239982	42076
GTCCTGCCGGTCTGGT	Sample	153915	152545	23151	129394	9158	120236	3521
CTTGATAGGTTCAACT	Sample	52702	52267	10993	41274	4403	36871	1307
ACGACTTCTAGATGAC	Sample	105685	105077	11097	93980	6023	87957	29058
TATACGGAAATAAGGT	Sample	101171	100335	13979	86356	5982	80374	19851
GACGATAAGTTCAGAA	Sample	36301	36016	4471				
ACGACTTCTTGGAGAG	Sample	72196	71883	5113				
GCCTGAACACCATTAT	Sample	146595	145265	20660				
AGCGTCTTTAGATGAC	Sample	42789	42592	710				
GGCCAGAGAATAAGGT	Sample	95225	94579	8858				
GACGATAATTCAAGCC	Sample	123513	122829	8731				
ACGTTATGTTGCGGAC	Sample	4177	4131	749				

Exon_Reads	Ambiguous_	Intron_Reads	Ambiguous_	Intergenic_R	Ribosomal_R	No_of_Genes
9711	189	47924	5438	44268	798	1413
84632	2341	78374	7235	67400	6544	2825
28801	1930	46001	5233	38271	1986	1836
4166	351	16436	2113	13805	395	680
38937	542	24231	2641	21606	6829	1105
31950	203	23448	2437	22336	3087	1279
4392	159	12328	1315	9690	822	529
9316	657	20957	1649	15387	540	1237
12656	926	52779	4658	43490	17	1611
36207	1723	818	37	728	2024	917
63511	4479	4442	542	5218	4581	903
59965	2981	21216	2094	19448	10474	1215
2845	193	44	4	37	10	616

Figure 18. Example stats file output.

C. Gene matrix file

The gene matrix file (sometimes called the gene table or counts matrix) is also in CSV format and contains gene counts for each barcode, with the genes in the rows and barcodes/cells in the columns. The file contains raw counts that can then be normalized and transformed accordingly using hanta. An example is shown below.

GeneID	CGTCGAGGT(	TCATACCAGT	TGGATCAAGT	TGGATCAAG	CATTCGGTT(	ATTCTACCCT	TATACGGAG	AACCGGTTT	...
ENSMUSG00000102693	211	884	858	254	100	906	687	767	
ENSMUSG00000064842	0	423	895	91	544	0	796	615	
ENSMUSG00000051951	770	176	118	117	0	857	851	945	
ENSMUSG00000102851	258	189	120	724	78	448	372	486	
ENSMUSG00000103377	950	897	336	710	687	847	170	0	
ENSMUSG00000104017	0	0	30	74	531	29	28	910	
ENSMUSG00000103025	388	446	912	179	0	995	587	850	
ENSMUSG00000089699	153	183	143	0	599	881	411	189	
ENSMUSG00000103201	39	756	794	730	268	133	181	529	
ENSMUSG00000103147	257	347	0	604	217	8	479	369	
ENSMUSG00000103161	770	898	282	19	756	519	42	624	
ENSMUSG00000102331	421	279	257	273	88	227	0	0	
ENSMUSG00000102348	0	985	775	302	194	803	156	527	
...									

Figure 19. Example of a gene matrix file.

D. Gene info file

The gene info file is a CSV-formatted file that contains the main annotation for the genes as described in the GTF file that is part of the genome build. It includes gene ID, gene symbol, and gene length (used for some normalizations).

Gene_ID	Gene_Name	Gene_Biotype	Gene_Length	
ENSG00000187634	SAMD11	protein_codin	4166	
ENSG00000188976	NOC2L	protein_codin	5539	
ENSG00000187961	KLHL17	protein_codin	3395	
ENSG00000187583	PLEKHN1	protein_codin	2833	
ENSG00000187642	PERM1	protein_codin	3424	
ENSG00000284332	MIR1302-2	miRNA	138	
ENSG00000237613	FAM138A	lincRNA	1219	
ENSG00000268020	OR4G4P	unprocessed_	840	
ENSG00000186092	OR4F5	protein_codin	2618	
ENSG00000238009	AL627309.1	lincRNA	3726	
ENSG00000239945	AL627309.3	lincRNA	1319	
		...		

Figure 20. Example of a gene info file.

E. R Data objects

During the generation of the HTML report described above, RData objects are created with results of various modules. These objects can be used directly as input into the interactive hanta R kit to perform further analysis. This also saves processing time.

Please refer to the hanta R kit User Guide for details.

Appendix A. Troubleshooting

If you encounter errors using the mappa pipeline, please capture a screenshot or the text of the error you may be seeing on the screen and send that plus the relevant log file to technical_support@takarabio.com.

Table II. Potential issues encountered with mappa software and the log files related to that area.

Problem area	Log filename	Directory location
Installation	mappa_install.log	%MAPPA_HOME
Demuxer	demux_mappa_demuxer.log	[Configured demux output folder]*
Analyzer	analysis_mappa_analyzer.log	[Configured analysis output folder]*

* - See Figure 21, below

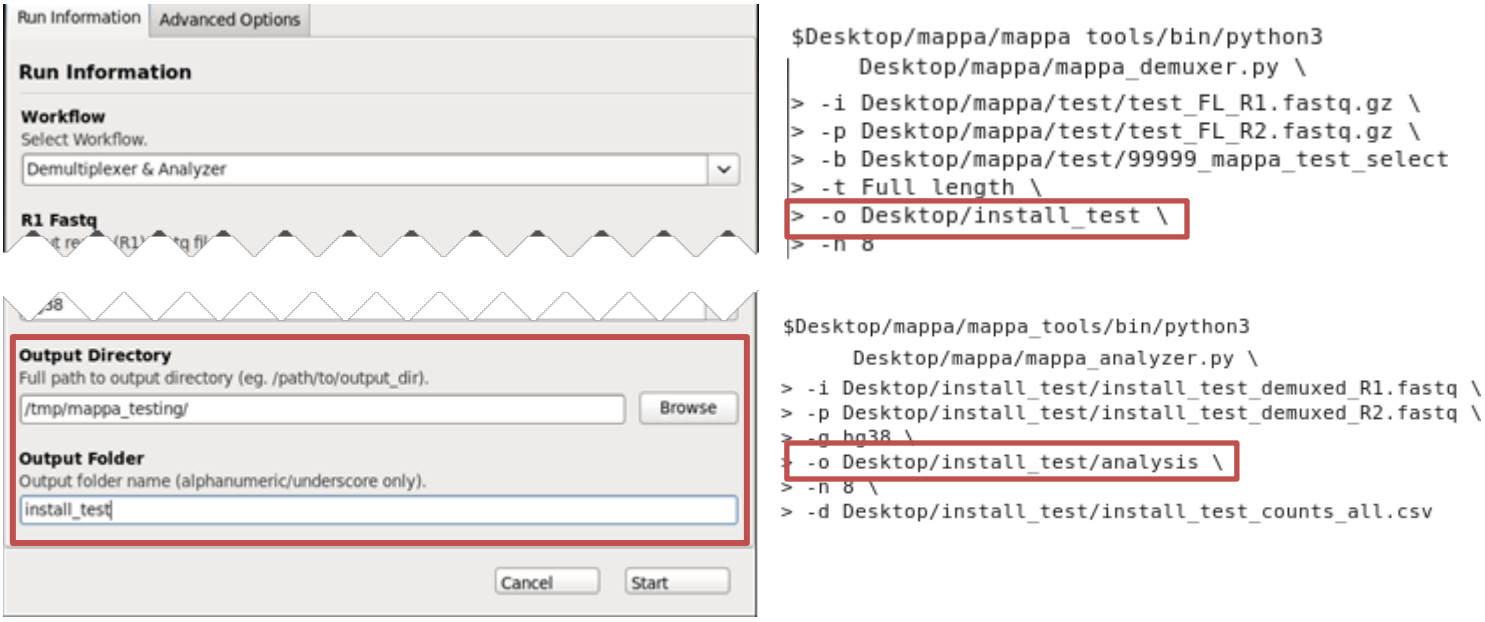


Figure 21. Locations to find the demux and analysis output folders from the UI configuration (left) and command line (right).

Appendix B. Remote access to the Linux server

The steps below provide a high-level walk through for setting up remote access to the Linux server through a remote access program. See Section II.D for more information about potential remote access programs that could be used.

The example used below shows the VNC software, but this would be similar for other programs.

1. Install the remote access server software on the Linux server. This requires superuser/root access, so contact the server administrator if you do not have that level of permission or access to the server.
2. On the desktop computer from which the server will be accessed, such as a laptop or workstation, install the remote access client. Administrator access is not required by default, but please contact your local IT group if problems are encountered.

   Software and details for download can be found at the links provided in Section II.D.

3. Once installed, launch the remote access client.
4. Enter the server's name or IP address as shown in Figure 22, below.

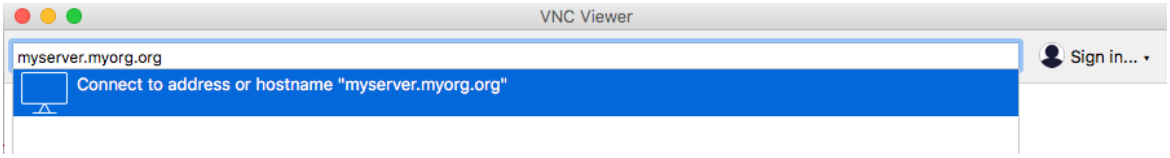


Figure 22. Example of inputting the address name of the server into the remote access client (VNC) to connect to it.

5. A user account is required on the Linux server to log in. Once logged in, you might see a screen like Figure 23 (below).

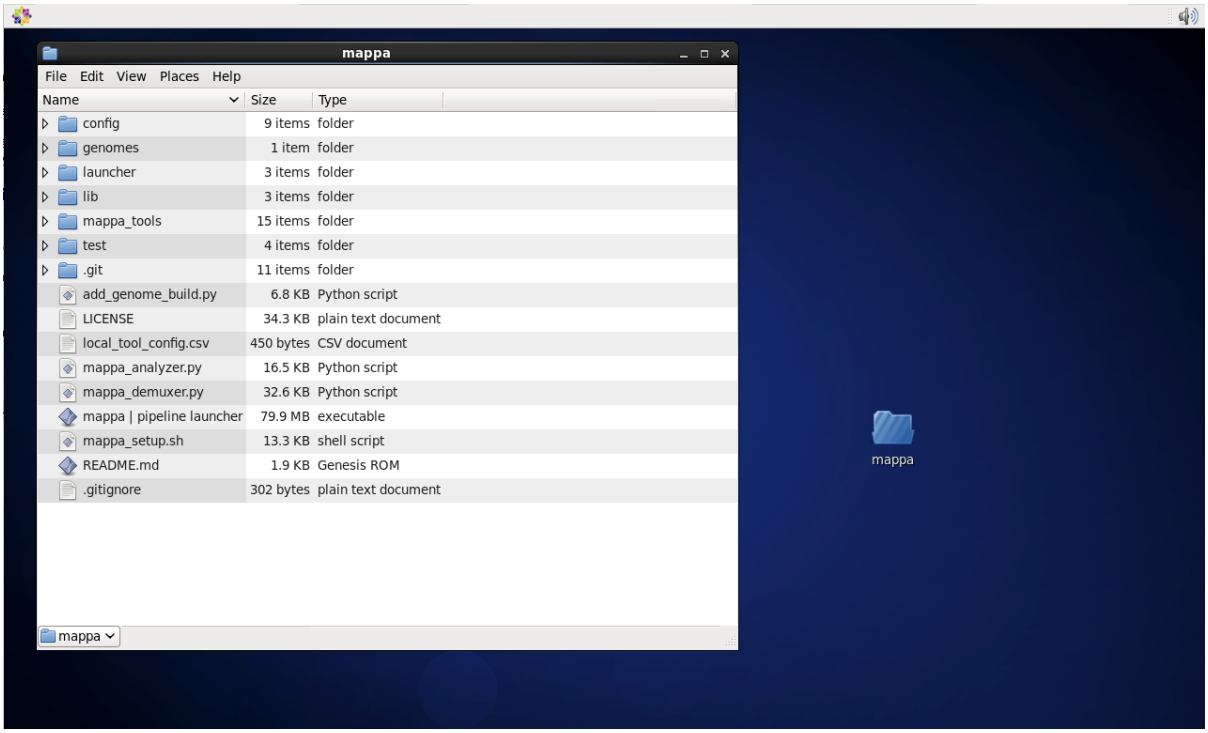


Figure 23. Example screenshot of the UI while accessing the Linux server remotely through a desktop client, after connecting and logging in.

6. Proceed with running mappa Analysis Pipeline (Section V).

> **NOTE**: Issues connecting to the server or with the login credentials should be directed to your local IT group or the server admin for the Linux server.

Contact Us	
Customer Service/Ordering	**Technical Support**
tel: 800.662.2566 (toll-free)	tel: 800.662.2566 (toll-free)
fax: 800.424.1350 (toll-free)	fax: 800.424.1350 (toll-free)
web: takarabio.com	web: takarabio.com
e-mail: ordersUS@takarabio.com	e-mail: technical_support@takarabio.com

Notice to Purchaser

Our products are to be used for **Research Use Only**. They may not be used for any other purpose, including, but not limited to, use in humans, therapeutic or diagnostic use, or commercial use of any kind. Our products may not be transferred to third parties, resold, modified for resale, or used to manufacture commercial products or to provide a service to third parties without our prior written approval.

Your use of this product is also subject to compliance with any applicable licensing requirements described on the product's web page at takarabio.com. It is your responsibility to review, understand and adhere to any restrictions imposed by such statements.

Takara Bio USA, Inc.

United States/Canada: +1.800.662.2566 • Asia Pacific: +1.650.919.7300 • Europe: +33.(0)1.3904.6880 • Japan: +81.(0)77.565.6999

12.19 US