Pipeline Setup and Use (Current Version)

1 month ago · Updated

***NOTE**: This document is for the current pipeline version **>=v1.0.4**. Documentation for earlier versions can be found here: Pipeline Setup and Use (v1.0.0-1.0.3).*

Version **>=v1.0.4** supports data generated with v1 and v2 chemistry from the WT Mini, WT, and WT Mega kits, Evercode TCR, and Gene Capture.

Outline

Overview

Parse's single cell kits facilitate scalable single cell RNA-seq, enabling you to go all the way from samples in a single cell suspension to sequencing data. This document covers the deployment, set up and use of a bioinformatics pipeline to process the resulting data. The inputs to the pipeline consist of fastq files generated after sequencing single cell libraries, and additionally a gene list if you are doing Gene Capture. Outputs include summary reports and data files relevant to the data type.

To effectively use the data analysis software described, several prerequisites are required of users. For regular bioinformatics practitioners there should be nothing out of the ordinary; users should be comfortable with Linux command-line tools and system configuration commands. Pipeline software is written in Python 3 and uses Pandas, Numpy, Scanpy and some plotting and clustering libraries. For genome alignments, STAR and Samtools are used. IgBlast is used in TCR analysis. Building on the Miniconda Python platform means that many required software components may be installed as a single bundle. For the additional software dependencies, bash shell scripts are provided. Dependency installation may be limited to an Anaconda environment if sys admin privileges are not available, or installation may be system wide with appropriate permissions (e.g. sudo).

Note about terminal commands in this document:

- Terminal commands are shown in `mono-space font`. In some cases, long commands may appear to wrap across multiple lines; This is a formatting artifact. In these cases, simple cut-and-paste of commands from this document to a terminal may not work.
- The tag `NOTE: Single-line-command` is used before commands that may appear wrapped (but should actually be a single command line).
- If single commands *must* be wrapped across multiple lines (e.g. in a script), make sure that lines end with a slash `'\'` character (Avoid any additional characters, even spaces, after such slashes, as this may be problematic).

Set up software environment

Using AWS (Amazon Web Services)

For brevity and simplicity, details presented here cover a single workflow environment and process. Described analysis steps were tested with a (newly spawned) Amazon Web Services (AWS) cloud-based compute instance; other compute configurations (i.e. multi-purpose, on-site compute farm arrangements) should work as well.

Instructions to set up AWS for use with the pipeline can be found in a separate document, Amazon Web Services (AWS) Setup.

Install and activate Miniconda

Miniconda (the minimal version of "Anaconda") is a software distribution that contains a package manager (referred to as "conda") which allows one to install software in their home directory *without* administrative (i.e. sudo) privileges. This step is also necessary to ensure that the appropriate versions of pipeline dependencies are installed and do not conflict with the user's current software environment. Installing conda is fairly straightforward; to do so please follow these steps:

- In your terminal **home** directory, type
  - `wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh`
- After the install script downloads, execute the script by prepending the script with the `bash` command:
  - `bash Miniconda3-latest-Linux-x86_64.sh`
- As the script proceeds through the installation process, hit **"Enter"** or **"Yes"** when prompted.

Regardless of install process followed, make sure conda is *activated* before proceeding. Normally, activation is the last step of the installation. If conda is active in a terminal, the command `conda --version` will report a number and you will see and `(base)` next to your shell prompt; if the command instead reports an error, try `source ~/.bashrc` to activate and check again. `conda` can also be enabled by running `conda activate` and disabled by `conda deactivate`.

Create a new conda environment

Next, create a new conda environment with Python 3.9.

```
# Create new environment with Python 3.9
conda create -n spipe python=3.9

# Activate your new environment
conda activate spipe
```

Suggested directory structure

To help manage files for genomes, sequencing data, and analysis, we suggest something like the directory structure shown below. If following the AWS instructions, our EBS "disk" will be accessible to our EC2 as `/newvolume`. The file structure suggested is directly under this partition. An example layout is this:

```
newvolume
|-- analysis
|   |-- 202206_ex1_an1
|   |-- 202206_ex1_an2
|   |-- 202206_ex1_an3
|   \-- 202206_ex1_comb
|-- expdata
|   \-- 202206_ex1
\-- genomes
    |-- hg38
    \-- hg38_mm10
```

- The `/newvolume/analysis/` directory is intended to hold pipeline outputs. Subdirectories are created by the pipeline when it is run. The example shows four subdirectories, corresponding to four pipeline runs (Actual pipeline output directories have deeper, nested structures; This is not shown above for clarity).
- The `/newvolume/expdata/` directory is intended to hold experimental data (i.e. sequencing fastq files). Sequencing output from different experiments (e.g. fastq files) may be placed into different subdirectores.
- The `/newvolume/genome/` directory is for reference genome files. The example has subdirectories for human-only and human-mouse mixed species experiments.
- Below commands will create the top-level part of the above directory structure.

```
cd /newvolume/
mkdir analysis expdata genomes
```

Installing the Parse Biosciences pipeline

- The pipeline software and associated data files are obtained as a zip file package. See the Pipeline Download (Current Version) page.
- Expansion of the zip archive may be done anywhere on your target system (e.g. EC2), as the zip file and contents are not normally needed after the pipeline is installed. For the example below, the zip archive is expanded to create a new subdirectory just below the top level.
- NOTE. You may need to install the `unzip` program to decompress zip files if this is not present. This can be done via conda or apt (e.g. `conda install unzip`)

```
# Unzipping zip package will create new subdirectory under /newvolume
unzip ParseBiosciences-Pipeline.1.0.5p.zip -d /newvolume
```

- Setup is done in the newly expanded directory.

```
# Change into the package directory
cd /newvolume/ParseBiosciences-Pipeline.1.0.5p/
```

- Software dependencies may be installed via one of two routes. ***IMPORTANT:*** *Follow only* ***one*** *of the two options to install software dependencies. DO NOT run both install scripts, as this can cause potential version conflicts that may be non-obvious and non-trivial to correct.*
- **OPTION 1)** If **sys admin privileges are not available**, installation may be limited to an Anaconda environment. In this case, run the 'install_dependencies_conda.sh' script (To simply list, but not install any packages, use command argument '-d')

```
# OPTION 1; Does NOT require sudo privileges
# Anaconda environment install script in pipeline directory
bash ./install_dependencies_conda.sh -i -y
```

- **OPTION 2)** Alternatively, if sys admin privileges are available (e.g. sudo permission), installation may be done at the system level via the 'install_dependencies.sh' script (To simply list, but not install any packages, use command argument '-d')

```
# OPTION 2; Requires sudo privileges
# System-wide install script in pipeline directory
bash ./install_dependencies.sh -i -y
```

- Pipeline package install uses pip. ***IMPORTANT****: Regardless of which installation script is used, anaconda must be activated in the terminal before issuing the pip command; If unsure, see Anaconda installation step above.*

```
# pip runs the script 'setup.py' in the pipeline directory
pip install --no-cache-dir ./
```

- To check the installation, call the script without any arguments. This reports a minimal message (but no errors).

```
# Call without arguments
split-pipe
```

The further commands display additional information (somewhat lengthy output)

```
# Displays usage screen
split-pipe --help

# Briefly describes processing steps, option formatting
split-pipe --explain
```

Installing TCR dependencies

NOTE: *Pipeline version >= 1.0.4p is required for running TCR analysis.*

- To install TCR-specific dependencies, the following commands are run from the directory where the Parse Biosciences pipeline is installed.
- ***IMPORTANT****: After the install_TCR.sh script has completed, the pipeline install step (i.e. 'pip install --no-cache-dir ./') must be run again.*

```
# Change into the package directory
cd /newvolume/ParseBiosciences-Pipeline.1.0.5p/
bash ./install_TCR.sh -i -y
```

```
# rerun pip install step to setup pipeline with TCR analysis
pip install --no-cache-dir ./
```

It is also possible to check whether TCR analysis is already installed using the following commands.

```
cd /newvolume/ParseBiosciences-Pipeline.1.0.5p/
bash ./install_TCR.sh -c
# Checking TCR status
# TCR appears to be installed; Found 207 files
```

Generating an indexed reference genome

To run the pipeline, you will first have to generate an indexed genome that can later be used to align reads during processing of single cell libraries.

You will need to start with reference sequence (.fasta) and gene annotation (.gtf) files for each species to be processed. These files are normally gzip compressed (.gz). As suggested above, we will put these files in a specific subdirectory.

```
cd /newvolume
# Create genome dir (If not already created earlier)
mkdir genomes
```

Download files: In this example we will download the fasta and gtf from Ensembl using the wget command. Note that the genome file versions shown below may be updated.

```
# Go to the new genomes dir
cd /newvolume/genomes

# Download via wget
# NOTE: Single-line-command; Each ‘wget xyz’ is a single command
#   It may be easiest to create a simple script file with each
#   line holding a wget command with appropriate filenames for
#   the genomes of interest. Then run ‘bash <filename>’
wget https://ftp.ensembl.org/pub/release-109/fasta/homo_sapiens/dna/Homo_sapien
wget https://ftp.ensembl.org/pub/release-109/gtf/homo_sapiens/Homo_sapiens.GRCh
```

```
wget https://ftp.ensembl.org/pub/release-109/fasta/mus_musculus/dna/Mus_musculu
wget https://ftp.ensembl.org/pub/release-109/gtf/mus_musculus/Mus_musculus.GRCm
```

Use the `split-pipe --mode mkref` command to create processed reference sequence index files. Required input arguments specify run mode, genome names, sequences (fasta), gene annotations (gtf), and output directory.

Here is the call to make a human (only) reference:

```
# NOTE: Single-line-command -OR- multi-line with backslashes
# (as described above)
split-pipe \
--mode mkref \
--genome_name hg38 \
--fasta /newvolume/genomes/Homo_sapiens.GRCh38.dna.primary_assembly.fa.gz \
--genes /newvolume/genomes/Homo_sapiens.GRCh38.109.gtf.gz \
--output_dir /newvolume/genomes/hg38
```

Here is an example call (explained more below) to make a reference for mixed species (human and mouse) experiments. In this example, all of the output files will be created in the `/newvolume/genomes/hg38_mm10/` directory.

Note: A species mix experiment is where cells from two different species are *explicitly mixed* and placed into the same well in a round 1 plate. This is a use case where a combined genome is ideal. This *should not* be confused with an experiment where multiple species are placed into *separate* wells as different samples. In the latter scenario, it's recommended to create an individual reference for each species of interest to minimize reads lost due to multi-mapping. This is explained in further detail in the section "Running the pipeline for one sublibrary" under "***Note:*** *Species mix vs. species separate samples*".

```
# NOTE: Single-line-command -OR- multi-line with backslashes between
#   command line options (All options for given command flag must be
#   on a single line; e.g. both fasta or gtf files if inputting two
#   genomes as the example below)
#   As mentioned above, it may be easier to create a simple script
#   file holding the full command line (e.g. below) rather than
#   actually typing everything at the command line.
```

```
split-pipe \
--mode mkref \
--genome_name hg38 mm10 \
--fasta /newvolume/genomes/Homo_sapiens.GRCh38.dna.primary_assembly.fa.gz \
        /newvolume/genomes/Mus_musculus.GRCm39.dna.primary_assembly.fa.gz \
--genes /newvolume/genomes/Homo_sapiens.GRCh38.109.gtf.gz \
        /newvolume/genomes/Mus_musculus.GRCm39.109.gtf.gz \
--output_dir /newvolume/genomes/hg38_mm10
```

Explanation of arguments to split-pipe:

- `--mode mkref` argument directs the pipeline to make a processed reference genome.
- `--genome_name <genome name(s)>` specifies that the names for each genome used to make the reference. In the example above, two genomes (hg38 and mm10) have been specified since this was a mixed-species experiment.
- `--fasta <path to fasta(s)>` specifies the reference genome fasta sequences to be used. These should correspond to the named genomes (and must be ordered correctly).
- `--genes <path to gtf(s)>` specifies the reference gene annotations (in GTF format) to be used. These should match the named genomes and reference fasta files (and must also be ordered correctly).
- `--output_dir <output path>` specifies the directory to put all of the output files for the indexed genome.

Note. When preparing a reference genome, STAR will output a log file that can be quite large. This is not normally used and may be deleted to save space (e.g. `rm Log.out`).

Using the pipeline to process sequencing data

Processing data from Parse's kits will involve 3 steps (after an appropriate indexed reference genome has been created):

- Generating fastq files from the Illumina bcl files (using bcl2fastq)
- Processing the data from each sublibrary individually (using `split-pipe --mode all`)
- Combining the processed data from each sublibrary into a single dataset (using `split-pipe --mode comb`)

Pipeline options for processing data from each of Parse's kits (WT Mini, WT, or WT Mega) are normally the same, with kit identity determined automatically from the data. For this to work, the

chemistry version associated with the data must be explicitly provided as a pipeline argument. All kits purchased after September 1st, 2022 are v2 chemistry.

Generating fastq files using bcl2fastq

The first step in processing your data after sequencing is generating fastq files for each sublibrary. Detailed directions on how to do this using bcl2fastq can be found here.

IMPORTANT: If you have fastq files from multiple lanes they must be concatenated, but sublibraries should always remain separate. For example:

```
cat S1_R1_L001.fastq.gz S1_R1_L002.fastq.gz > S1_R1_cat.fastq.gz
```

Running the pipeline for one sublibrary

Sequencing data from each sublibrary should first be run through the analysis pipeline separately (the steps to combine sublibraries into a single dataset is described below).

To illustrate pipeline use for analysis, we will go through an example dataset produced from a species-mixing experiment with both mouse and human cells mixed in the same well. In this example, bcl2fastq has already been run and the output fastq files are found in the following subdirectory:

```
ls -c1 /newvolume/expdata/202206_ex1
s1_S1_R2_001.fastq.gz
s1_S1_R1_001.fastq.gz
```

During analysis, the `split-pipe` script will identify barcodes (and correct any sequencing errors), align sequencing reads, assign transcript counts to genes, generate a cell-by-gene count matrix, and provide process summary statistics and plots in an html report.

Required pipeline inputs specify run mode, chemistry version, sequencing (fastq) files, genome directory, output directory, and optional sample definitions. In this context, 'samples' are named subsets of wells.

Sample definitions consist of sample names and subsets of wells. Samples may be specified to the pipeline via several different mechanisms. They may be given explicitly as command line arguments. They may be specified via a simple text file list. Or they may be specified via a SampleLoadingTable excel file.

Specifying samples explicitly. Below is an example pipeline call (explained more below) to analyze mixed human-mouse sequencing data obtained using the 48 well WT kit. In this example, three samples with six wells each are specified explicitly via the command line.

Note: All kits purchased after September 1st 2022 use "v2" chemistry. Kits purchased before this date are "v1". For pipeline versions >= 1.0.0, the kit type will automatically be determined based on the chemistry argument.

```
# NOTE: Single-line-command -OR- multi-line with backslashes (as described abov

# Example with explicit sample definitions
split-pipe \
    --mode all \
    --chemistry v2 \
    --genome_dir /newvolume/genomes/hg38_mm10/ \
    --fq1 /newvolume/expdata/202206_ex1/s1_S1_R1_001.fastq.gz \
    --fq2 /newvolume/expdata/202206_ex1/s1_S1_R2_001.fastq.gz \
    --output_dir /newvolume/analysis/202206_ex1_an1 \
    --sample xcond_1 A1-A6 \
    --sample xcond_2 C1-C6 \
    --sample xcond_3 D1-D6
```

Explanation:

`--mode all` argument directs the pipeline to perform all process analysis steps for data from a single sublibrary (i.e. pre-process, align, post-process, generate expression matrices and reports).

`--chemistry <chemistry>` argument is required to specify the chemistry version. Providing the chemistry allows the pipeline to automatically determine the kit from the data (among other things). Valid chemistry values are v1 and v2.

`--genome_dir` argument specifies the path to the indexed genome directory (mixed human-mouse in this example).

`--fq1` and `--fq2` specify the path to the fastq files containing read 1 (includes cDNA) and read 2 (includes barcodes), respectively.

Note: If fastq files for read 1 and read 2 differ only by "R1" *vs* "R2" in the names, it is not necessary to specify `--fq2`. The R2 filename will be derived from R1.

`--output_dir` specifies the output directory. This is created by the pipeline and populated with output files.

`--sample <sample name> <wells>` specifies the sample name and corresponding wells in which each sample was loaded into the first round of barcoding. Each of these lines consists of the sample command line flag followed by a name and a specification of which wells to include. To see formatting rules for specifying well subsets, call `split-pipe --explain`. Only valid wells for the chosen kit are allowed. By default, a combined analysis of all wells (named 'all-well') is also performed by the pipeline.

Note: *Species mix vs. species separate samples.* The above example illustrates a pipeline run where cells from two species are mixed as part of the same sample (i.e. mixed in the same well). Species separate samples, on the other hand, are placed in separate wells and should be run as individual pipeline calls with their respective genome. This is done to avoid losing reads through "multi-mapping", a scenario where a read is removed due to aligning to more than one locus. Using a combined genome from two closely related organisms increases the probability that a read will map to more than one locus.

For species separate samples, it's advised to change the pipeline "minimum map fraction" to lower than 20% in case the experiment contains a small percentage of reads for a specific organism. To do this, use the following steps:

```
# Generate parameter file
echo "post_min_map_frac 0.01" > parfile.txt

# Add this parameter to your spilt-pipe "--mode all" call
--parfile parfile.txt
```

Note: To see a list of valid kit names and chemistry versions, run the pipeline with command line flag `--kit_list`. This causes the pipeline to report the below information then stop:

```
# Get listing of valid kit names
split-pipe --kit_list

# Output:
There are 3 installed kits:

    WT_mini      12 wells,   Chemistry: v1, v2
    WT           48 wells,   Chemistry: v1, v2
    WT_mega      96 wells,   Chemistry: v1, v2
```

Specifying samples via list. Instead of specifying samples via multiple `--sample <sample name> <wells>` command line arguments, sample specifications may be put into a simple, space delimited text file.

```
# Example with list of sample specifications

# List file contents:
cat sample-list.txt
xcond_1 A1-A6
xcond_2 C1-C6
xcond_3 D1-D6

# Call same as above but with sample list file, and only fq1
split-pipe \
    --mode all \
    --chemistry v2 \
    --genome_dir /newvolume/genomes/hg38_mm10/ \
    --fq1 /newvolume/expdata/202206_ex1/s1_S1_R1_001.fastq.gz \
    --output_dir /newvolume/analysis/202206_ex1_an1 \
    --samp_list sample-list.txt
```

Specifying samples via SampleLoadingTable. Samples may also be specified using the SampleLoadingTable excel file.

```
# Example with SampleLoadingTable

# Call same as above but using excel file
split-pipe \
    --mode all \
    --chemistry v2 \
    --genome_dir /newvolume/genomes/hg38_mm10/ \
    --fq1 /newvolume/expdata/202206_ex1/s1_S1_R1_001.fastq.gz \
    --output_dir /newvolume/analysis/202206_ex1_an1 \
    --samp_sltab SampleLoadingTable.xlsm
```

Note. To check that samples have been specified correctly before actually running the pipeline, the `--dryrun` command line option can be used. In this case, the pipeline will initialize everything and report status, including sample specifications, but then stop.

Running the pipeline for Evercode TCR data

TCR Sequencing data from each sublibrary should first be run through the analysis pipeline separately (the steps to combine sublibraries into a single dataset is described below).

Analysis with a paired whole transcriptome (parent) data

If the experiment was performed together with a whole transcriptome (parent) sequencing run, the results from this run are used to filter cells in the TCR analysis. Therefore, the whole transcriptome data needs to be analyzed first following the above instructions for running the pipeline for one sublibrary.

For illustration, pipeline outputs from the parent data runs (i.e. whole transcriptome sublibraries matched to TCR sublibraries) are as shown:

```
ls -c1 -d /newvolume/analysis/202301_tcr_ex1/WT/
S2/
S1/
```

To illustrate pipeline use for TCR analysis, we will go through an example dataset. In this example, bcl2fastq has already been run and the TCR fastq files required as input are found in the following subdirectory:

```
ls -c1 /newvolume/expdata/202301_tcr_ex1/TCR/
s2_S2_R2_001.fastq.gz
s2_S2_R1_001.fastq.gz
s1_S1_R2_001.fastq.gz
s1_S1_R1_001.fastq.gz
```

As with whole transcriptome data, the pipeline will run QC on inputs and correct barcodes (e.g. to correct sequencing errors). Next, for TCR analysis, the pipeline builds consensus contigs for each cell, annotates the contigs, generates clonotype summaries and provides process summary statistics and plots in an html report.

Below is an example pipeline command to run the TCR data alongside a parent (whole transcriptome) dataset. Note that the parent dataset has already run to completion.

```
split-pipe \
    --mode all \
    --chemistry v2 \
    --tcr_analysis \
    --parent_dir /newvolume/analysis/202301_tcr_ex1/WT/S1 \
    --output_dir /newvolume/analysis/202301_tcr_ex1/TCR/S1 \
    --fq1
/newvolume/expdata/202301_tcr_ex1/TCR/s1_S1_R1_001.fastq.gz \
    --fq2
/newvolume/expdata/202301_tcr_ex1/TCR/s1_S1_R2_001.fastq.gz
```

Note:

In addition to the commands described above `--parent_dir` argument lets the pipeline know where to find the WT results of the sublibrary. This is the directory with the outputs of WT pipeline for the sublibrary. This argument is necessary to filter TCR results. If this argument is not given, only unfiltered results will be generated.

Analysis without a paired whole transcriptome (parent) data

It is possible to run TCR analysis without a paired whole transcriptome (parent) data. In this case the `--parent_dir` parameter will not be provided. This also means that no filtering will be performed and only unfiltered results will be generated.

When the TCR analysis is done without a parent, required pipeline inputs for TCR analysis are run mode, chemistry version, a flag to indicate TCR analysis, sequencing (fastq) files, and an output directory. In the absence of a parent, samples (i.e. subsets of wells) can be specified in the same way as for non-TCR analysis. Sample specification may only be done if TCR analysis is done without a parent; If `--parent_dir` is provided, samples are automatically defined to match that.

Below is an example call to analyze the example data without specifying any samples and specifying only `--fq1`.

```
split-pipe \
    --mode all \
```

```
    --chemistry v2 \
    --tcr_analysis \
    --output_dir /newvolume/analysis/202301_tcr_ex1/TCR/S1 \
    --fq1 /newvolume/expdata/202301_tcr_ex1/TCR/s1_S1_R1_001.fastq.gz
```

Running the pipeline for Gene Capture data

To run the pipeline on Gene Capture samples, a list of the target genes is required. The same steps as above for running the pipeline for one sublibrary should be followed, with one additional argument
`--targeted_list` which specifies the target gene list.

The target gene list should contain a list of gene names in the target panel and **must match the gene names or ids as specified by the genome in the mkref step**. Extracting the header (first) line and a subset of gene records from the all_genes.csv file in the genome output directory is a good way to create a target gene list.

To illustrate pipeline use for Gene Capture analysis, the following commands include an example of a Gene Capture experiment in human samples. In this example, bcl2fastq has already been run and the output fastq files and target gene list are found in the following subdirectory:

```
ls -c1 /newvolume/expdata/202301_ex1
s1_S1_R2_001.fastq.gz
s1_S1_R1_001.fastq.gz
gene_panel.csv
```

In this example, two samples with 24 wells each are specified explicitly via the command line. Samples may also be specified explicitly, via a simple text file list or a SampleLoadingTable excel file as described in the running the pipeline for one sublibrary instructions above.

```
# Example with explicit sample definitions
split-pipe \
    --mode all \
    --chemistry v2 \
    --genome_dir /newvolume/genomes/hg38/ \
    --targeted_list /newvolume/expdata/gene_panel.csv \
    --fq1 /newvolume/expdata/202301_ex1/s1_S1_R1_001.fastq.gz \
    --fq2 /newvolume/expdata/202301_ex1/s1_S1_R2_001.fastq.gz \
    --output_dir /newvolume/analysis/202301_ex1_an1 \
```

```
    --sample example_1 A1-D6 \
    --sample example_2 A7-D12
```

Combining outputs from multiple sublibraries

After running each sublibrary through the pipeline, the outputs of each sublibrary can be combined into a single dataset using `split-pipe --mode combine`. For TCR analysis sublibraries, the `--tcr_analysis` option is also required. In the case where TCR sublibraries have matched parent outputs, both the parent and TCR sublibraries need to be separately combined; The parent outputs should be combined *before* attempting to combine corresponding TCR outputs and they should combine the same set of sublibraries respectively (see TCR-specific example using combine mode).

Note: *You can only combine libraries from the same library preparation using* `--mode combine`. *For example, you can combine multiple sublibraries from a WT kit, however you can not combine a WT sublibrary with a Gene Capture sublibrary. For information on how to integrate different kinds of datasets, please use our tutorial on how to analyze and integrate data with Scanpy and Harmony or read more about integration in Seurat here.*

To illustrate pipeline use for combining the outputs of multiple runs, assume we have previously generated outputs for three separate WT sublibraries in the following directories (as illustrated in the above suggested directory structure section). These are under the /newvolume/analysis directory and all have names containing 'ex1_an':

```
ls /newvolume/analysis | grep ex1_an
202206_ex1_an1
202206_ex1_an2
202206_ex1_an3
```

To combine previously generated output data, `split-pipe --mode combine` merges data and statistics files into new outputs and generates a new summary report describing merged data. A new output directory is created to hold the combined results.

Required pipeline inputs specify the run mode (i.e. 'combine'), a list of existing pipeline output directories for the sublibraries to be combined, and an output directory. For combine mode, chemistry, kit, genome, and sample definitions **should not be specified**, as this information is parsed from the sublibrary directories that will be combined. (Any supplied command line arguments for these parameters will not be used). All unique sample names found in all supplied sublibrary directories are processed as new, combined samples.

Here is an example call (explained more below) to combine results from three processed WT sublibraries. In this example, we specify each sublibrary path explicitly.

```
# NOTE: Single-line-command -OR- multi-line with backslashes (as described abov
split-pipe \
    --mode comb \
    --sublibraries /newvolume/analysis/202206_ex1_an1 \
                   /newvolume/analysis/202206_ex1_an2 \
                   /newvolume/analysis/202206_ex1_an3 \
    --output_dir /newvolume/analysis/202206_ex1_comb
```

Explanation:

`--mode combine` directs the pipeline to combine outputs from supplied sublibrary output paths.

`--sublibraries` specifies a list of paths to the sublibrary outputs to be combined.

`--output_dir` specifies the output directory, which is created by the pipeline.

Rather than listing all sublibrary paths on the command line as shown above, paths may be put into a file that is then loaded by the pipeline. To run the pipeline in this way, generate a file with one path per line. For example, assume this file is named "sublibs.lis"

```
cat sublibs.lis
/newvolume/analysis/202206_ex1_an1
/newvolume/analysis/202206_ex1_an2
/newvolume/analysis/202206_ex1_an3
```

Then, call the pipeline in combine mode using the `--sublib_list` rather than the `--sublibraries` argument

```
# NOTE: Single-line-command -OR- multi-line with backslashes
# (as described above)
split-pipe
    --mode comb \
    --sublib_list sublibs.lis \
    --output_dir /newvolume/analysis/202206_ex1_comb
```

TCR-specific example using combine mode

To illustrate pipeline use for combining the outputs of multiple runs, we have previously generated results for two sublibraries as shown above and their respective results follow the below directory structure.

```
ls -c1 -d /newvolume/analysis/202301_tcr_ex1/TCR/
S2/
S1/
```

Here is an example call (explained more below) to combine results from the two sublibraries above. In this example, we specify each sublibrary path explicitly.

```
#With a parent
split-pipe \
--mode comb \
–-tcr_analysis \
--parent_dir /newvolume/analysis/202301_tcr_ex1/WT/combined \
--output_dir /newvolume/analysis/202301_tcr_ex1/TCR/combined \
--sublibraries /newvolume/analysis/202301_tcr_ex1/TCR/S1 \
               /newvolume/analysis/202301_tcr_ex1/TCR/S2

#With no parent
split-pipe \
--mode comb \
–-tcr_analysis \
--output_dir /newvolume/analysis/202301_tcr_ex1/TCR/combined \
--sublibraries /newvolume/analysis/202301_tcr_ex1/TCR/S1 \
               /newvolume/analysis/202301_tcr_ex1/TCR/S2
```

Explanation:

`--mode combine` directs the pipeline to combine outputs from supplied sublibrary output paths.

`--tcr_analysis` argument lets the pipeline know that the input data is TCR data and triggers the TCR analysis part of the pipeline.

`--parent_dir` argument lets the pipeline know where to find the WT results of the sublibrary.

`--sublibraries` specifies a list of paths to the sublibrary outputs to be combined.

`--output_dir` specifies the output directory, which is created by the pipeline.

Note: *you may also use the* `--sublib_list` *argument rather than the* `--sublibraries` *argument to combine TCR data as described above.*

Analysis pipeline outputs

Running the pipeline to process one sublibrary, or to combine multiple sublibraries, results in a new output directory containing a number of subdirectories and files.

Considering the single sublibrary example illustrated above, an analysis run will produce the directory structure shown below. Under the top level output directory, there is a 'process' subdirectory containing processing files, and a subdirectory for each specified sample (i.e.--sample) containing sample-specific data. An "all-well" sample output is also generated that includes data for all wells in the experiment.

```
newvolume
`-- analysis
    `-- 202206_ex1_an1
        |-- process
        |-- all-well
        |-- xcond1
        |-- xcond2
        `-- xcond3
```

Top level files normally include sample-specific html summary files (named "<sample-name>_analysis_summary.html"), an aggregated pipeline metrics file summarizing all samples ("agg_samp_ana_summary.csv"), the pipeline log file, and a zip archive ("all_summaries.zip") holding the html, aggregated statistics and log files.

Each sample subdirectory has additional structure containing various sample-specific files. Here are outputs corresponding to one sample:

```
newvolume
`-- analysis
    `-- 202206_ex1_an1
        `-- <sample-name>
            |-- DGE_unfiltered
            |   |-- DGE.mtx
            |   |-- all_genes.csv
            |   `-- cell_metadata.csv
```

```
            |-- DGE_filtered
            |   |-- DGE.mtx
            |   |-- all_genes.csv
            |   `-- cell_metadata.csv
            |-- report
            |   |-- analysis_summary.csv
            |   `-- <various diagnostic / data files>
            `-- figures
                `-- <png figure files; Same data as html reports>
```

Explanation of Files:

`<sample-name>_analysis_summary.html` contains summary statistics and plots for a given sample. This can be opened and viewed in any web browser.

`<sample-name>/report/analysis_summary.csv` contains pipeline metrics for the sample.

`DGE.mtx` is a sparse matrix with cell-gene counts. Each row corresponds to a cell and each column corresponds to a gene (see "*genes.csv*" file for names/gene-id of each column).

`all_genes.csv` contains the gene name, gene id, and genome for each column in DGE.mtx. This file is the same as the file in the reference genome dir.

`cell_metadata.csv` file contains information about each cell including the cell barcode, species, sample, well in each round of barcoding, and number of transcript/genes detected.

Filtered vs unfiltered subdirectories:

`<sample-name>_DGE_filtered` subdirectory contains only the cells that passed the pipeline's minimum transcript count threshold. This threshold is calculated automatically based on the distribution of transcripts identified per cell.

`<sample-name>_DGE_unfiltered` subdirectory contains all "cells" including those that did not pass the pipeline's minimum transcript count threshold. This would be a good starting point for analysis if you want to determine a minimum transcript (or gene) threshold on your own, rather than relying on the built-in filtering in the pipeline.

Note: *The pipeline output for Gene Capture libraries contain the same files as above, but include 'target' in the file name to specify the output is from a Gene Capture analysis. (i.e.,* `all_genes.csv` *file is named* `target_genes.csv` *).*

Further details of pipeline usage

Steps and pipeline command examples illustrated above should suffice for normal pipeline setup and use. However, it may be useful to specify additional pipeline settings and options in some cases. Details of these options are described in a separate document: Pipeline Options (Current Version).

Was this article helpful?

Yes No

2 out of 2 found this helpful

Have more questions? Please contact support.

Return to top ^

Recently viewed articles

TCR Analysis Pipeline Outputs (Current Version)

Pipeline Options (Current Version)

Pipeline Download (Current Version)

Bioinformatics Onboarding Video

Evercode Sample Loading Tables

Related articles

Pipeline Download (Current Version)

Pipeline Options (Current Version)

Bioinformatics Onboarding Video

Pipeline Setup and Use (v1.0.0-v1.0.3)

Amazon Web Services (AWS) Setup

Support Suite