STOmics

Stereo-seq ANALYSIS WORKFLOW SOFTWARE SUITE USER MANUAL

Software Version: V7.0

Manual Version: A8

REVISION HISTORY

Manual Version: A0
Software Version: V1.0.0
Date: Nov. 2021
Description: Initial release

Manual Version: A1
Software Version: V2.1.0
Date: Dec. 2021
Description:

- **New feature:** addition of manual registration function, some fine-tuning will be automatically performed after manually registering;
- **Improvement:** performance improvement for `mapping`; the order of sequencing saturation calculation has switched, in v2.1.0 we only compute the saturation of tissue-covered region;
- **Bug fix:** fixed the bug of indexing at `mapping` step; fix the bug of long waiting time at `register` step.

Manual Version: A1.1
Software Version: V2.1.0
Date: Jan. 2022
Description:

- Add error handling;
- Update demo output.

Manual Version: A2
Software Version: V4.1.0
Date: Apr. 2022
Description:

- **New feature:** support to process fused microscopic images; employ Stereopy in clustering; new gene expression matrix file format; including a file format convertor and a `mapping` memory estimator;
- **Improvement:** performance improvement for `mapping`; update gene annotation approach in `count`; update image stitching and tissue segmentation model for better performance on image stitching and segmentation for tissue with voids.

Manual Version: A3
Software Version: V5.1.3
Date: Aug. 2022
Description:

- **New feature:** addition of cell segmentation on microscopic image in `register` pipeline module and `cellCut` pipeline module to extract cell expression matrix; design IPR file to store image processing record information and TIFF images produced in `register`; output exon expression matrix along with total MID count; addition of cell clustering analysis; addition of cell bin statistic result and image information in HTML report; add pipeline modules to support single-end FASTQ data; addition of option for selecting multi-mapped reads; addition of score system in image processing;
- **Improvement:** addition of poly A filtration in `mapping`; performance improvement for `merge`, `tissueCut` and `saturation`; add data scaling and upgrade clustering analysis pipeline;
- **Bug fix:** fix the bug of plotting scatter plot in `tissueCut`, review and modify the ambiguous metrics names and explanations in HTML report.

Manual Version: A4
Software Version: V5.4.0
Date: Nov. 2022
Description:

- **New feature:** render a more elegant way of organizing SE FASTQs input into `mapping`; addition of header for `count` output TXT file; upgrade `ipr2img` to `imageTools` which allows you to merge TIFF images to check segmentation result and plot templates on the panoramic image or registered image to check the result of stitching and registration;
- **Improvement:** change data struct of gene index in the GEF file from uint16 to uint32 to store more genes;
- **Bug fix:** fix the typo in the manual.

Manual Version: A3.1
Software Version: V5.1.3
Date: Dec. 2022
Description:

- **Bug fix:** fix the typo in the manual.

Manual Version: A5
Software Version: V5.5.0 - V5.5.2
Date: Jan. 2023
Description:

- **New feature:** addition of `manualRegister` and `lasso` pipeline modules, which acquire parameters and essential files from offline visualization software **StereoMap**; addition of error code function in each module, and the explanation has been organized in appendix D;
- **Improvement:** upgrade `tissueCut` pipeline module for better performance and improved accuracy of tissue recognition directly from gene expression matrix; upgrade clustering analysis pipelines; addition of image layers in HTML report which allow users to switch images displayed with gene expression distribution plot; the stitching deviation heatmap(s) and the frame of axes in the HTML report have been adjusted according to the registration parameters.

Manual Version: A5
Software Version: V5.5.3
Date: Mar. 2023
Description:

- **Improvement:** upgrade `mapping` pipeline module, which performs polyA filtering after CID mapping, for improved statistical output in `report`.

Manual Version: A6
Software Version: V6.0
Date: Mar. 2023
Description:

- **New feature:** addition of `rRNAremove` switch in `mapping`, off by default, to count and filter rRNA reads using the reference genome with the addition of rRNA information. rRNA count and ratio are recorded in the output file; support the scenario with the combination of DAPI and mIF (multiple immunofluorescence images), correspondingly generating registered, tissue segmentation, and cell segmentation results; support continuously processing image files output from each module of ImageStudio;
- **Improvement:** reconstructed `tissueCut` module pipeline, with the addition of parameter`-t` which means the number of used CPUs, to greatly improve the calculation efficiency; modified parameter `-g` in `img2rpi` module to a two-layer structure, upgraded RPI file format to support storing and organizing multiple stained microscopy images in groups; displayed pseudo colors (up to 7 colors) on the bottom layer of clustering results in HTML report, when handling the scenario of mIF in `report` module;
- **Bug fix:** fix the bug caused by version compatibility when generating CGEF file.

Manual Version: A6.1
Software Version: V6.0.1 & V5.4.4
Date: Apr. 2023
Description:

- **Bug fix:** fix the bug that the execution path of `checkGTF` function was invalid.

Manual Version: A6.1
Software Version: V6.0.2
Date: Jun. 2023
Description:

- **Improvement:** removed the dependence of parameters related to stitched-image stitching-evaluation from different versions of QC. The evaluation of stitched-image stitching is completed by `register` module; prompt an error when <StainType> in `.ipr` file is '-'; supported Zeiss microscope-stitched image;
- **Bug fix:** for `register`, fixed the issue that the module was unable to get `.czi` file path; fixed the issue of doing cell segmentation in `rapidRegister` for manually-stitched data; fixed the issue that the registered `.ipr` file that could not be registered again; expanded the chip-prefix list and fixed the track line template acquisition error. For `imageTools`, fixed the issue that bin_100 of cellMask in the SN`.rpi` file did not save corresponding metadata; fixed the issue that the merge could not be performed on images with the same width and height.

Manual Version: A7
Software Version: V6.1
Date: Jul. 2023
Description:

- **New feature:** supported outputs of Valid ClD Reads and un-mapped reads in FASTQ format; supported QC-failed but manually processed images as inputs for subsequent workflow; addition of tissue area (nm^2) in corresponding GEF file; supported generating GEF file of labeled region; addition of quality control alerts and tissue segmentation display in `report`;
- **Improvement:** updated `mapping` module to improve computational efficiency; updated the error alerts on checking the length of the gene name, in `checkGTF` and `count` modules.

Manual Version: A7
Software Version: V6.1.1
Date: Aug. 2023
Description:

- **Bug fix:** for `manualRegister`, fixed the issue that obtaining information from manual registration file `.json` incorrectly resulted in wrong width and height of images; for `tissueCut`, corrected the statistical method of `Reads_under_tissue` item in `tissueCut`, which now represents "Number of reads with position prior to filtration under tissue coverage"; for `report`, compatible with the `.ipr` file which is failed with image quality control (QC) and has no record of heat map matrix information needed for html report.

Manual Version: A7
Software Version: V6.1.2
Date: Sep. 2023
Description:

- **Bug fix:** for `manualRegister`, fixed the offset calculation error of whcih the scale direction was incorrect.

Manual Version: A7
Software Version: V6.1.2
Date: Oct. 2023
Description:

- **Bug fix:** for `imgeTool`, fixed the problem that manually registered images were incorrectly cropped after scale operation; for `report`, fixed the missing percentage value of floating text box for 'Sequencing Saturation', in HTML diagram.

Manual Version: A8
Software Version: V7.0
Date: Nov. 2023
Description:

- **New feature:** supported H&E image in pipeline; addition of `cellCorrect` module for expanding cell boundaries; addition of `cellChunk` module for displaying cell expression heatmap in StereoMap; addition of new parameter in `spatialCluster` addition of a new parameter in spatialCluster for adjusting Leiden clustering resolution; addition of output GEF file in `lasso` module;
- **Improvement:** performance improvement in `count` and `register` modules; updated tissue and cell segmentation model for nuclei-stain image; supported for using GPU in `register` module to accelerate computing.

Note: Please download the latest version of the manual and use it with the software specific to this manual.

WORKFLOW

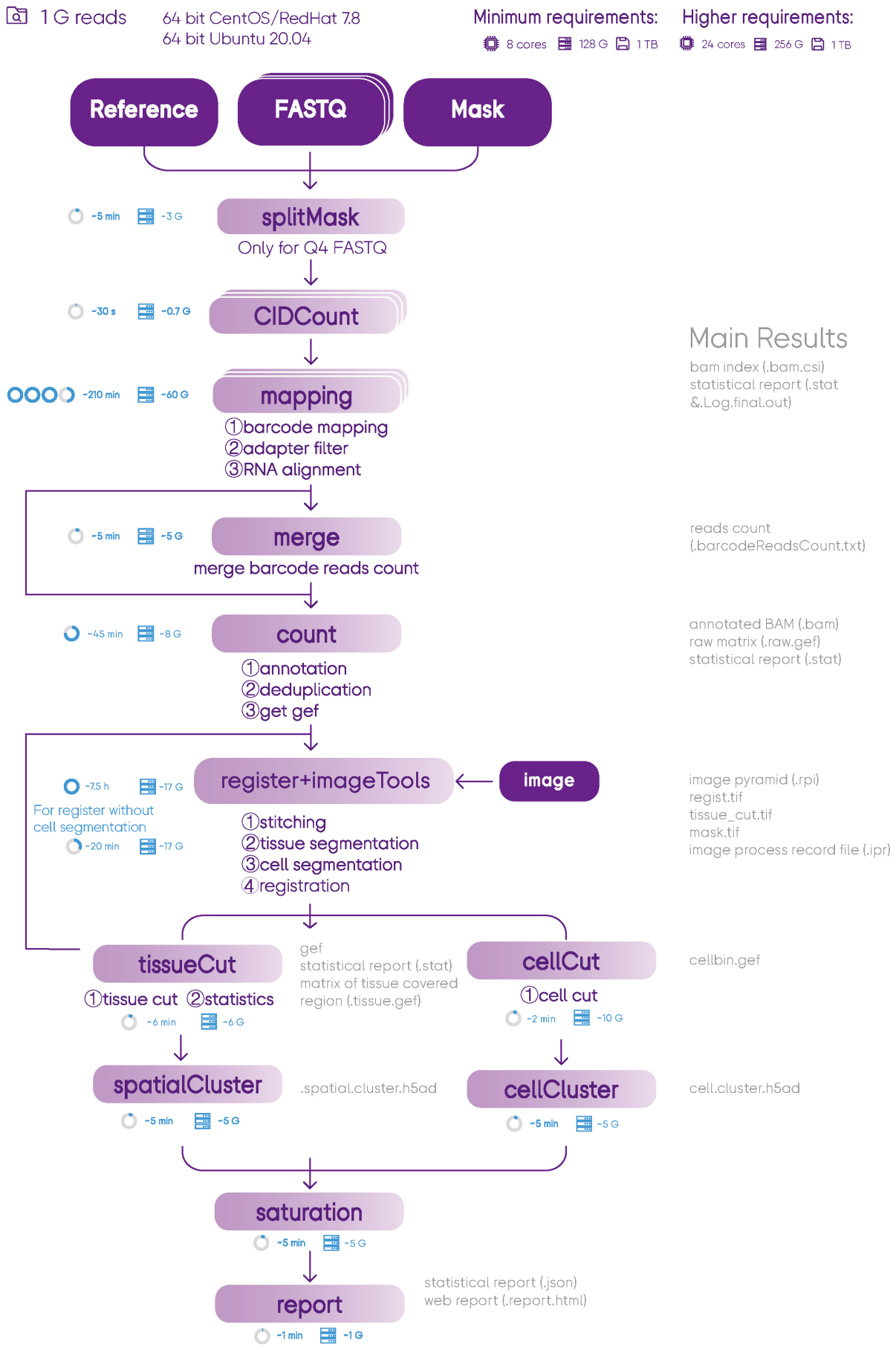

1 G reads
64 bit CentOS/RedHat 7.8
64 bit Ubuntu 20.04
Minimum requirements:
8 cores 128 G 1 TB
Higher requirements:
24 cores 256 G 1 TB
Reference
FASTQ
Mask
~5 min ~3 G
splitMask
Only for Q4 FASTQ
~30 s ~0.7 G
CIDCount
~210 min ~60 G
mapping
①barcode mapping
②adapter filter
③RNA alignment
~5 min ~5 G
merge
merge barcode reads count
~45 min ~8 G
count
①annotation
②deduplication
③get gef
~7.5 h ~17 G
For register without cell segmentation
~20 min ~17 G
register+imageTools
image
①stitching
②tissue segmentation
③cell segmentation
④registration
tissueCut
①tissue cut ②statistics
~6 min ~6 G
gef
statistical report (.stat)
matrix of tissue covered region (.tissue.gef)
cellCut
①cell cut
~2 min ~10 G
cellbin.gef
spatialCluster
~5 min ~5 G
.spatial.cluster.h5ad
cellCluster
~5 min ~5 G
cell.cluster.h5ad
saturation
~5 min ~5 G
report
~1 min ~1 G
statistical report (.json)
web report (.report.html)
Main Results
bam index (.bam.csi)
statistical report (.stat &.Log.final.out)
reads count (.barcodeReadsCount.txt)
annotated BAM (.bam)
raw matrix (.raw.gef)
statistical report (.stat)
image pyramid (.rpi)
regist.tif
tissue_cut.tif
mask.tif
image process record file (.ipr)

TABLE OF CONTENTS

APPENDICES

CRITICAL STEPS: Pay extra attention to these instructions/steps to avoid problematic results.

SOLUTION: Provides a solution to handling common errors.

CHAPTER 1
Stereo-seq ANALYSIS WORKFLOW SOFTWARE SUITE

1.1. Software Introduction

Stereo-seq Analysis Workflow[1] (SAW) software suite is a set of pipelines bundled to position sequenced reads to their spatial location on the tissue section, quantify spatial gene expression and visually present spatial expression distribution. SAW processes the sequencing data of Stereo-seq[2] to generate spatial gene expression matrices, and then users could take these files as the starting point to perform downstream analysis. SAW includes thirteen essential and suggested pipelines and auxiliary tools for supporting other handy functions:

- **splitMask:** Split Stereo-seq Chip T mask file into several pieces according to CID indexing in the Q4 FASTQ files.
- **CIDCount:** Count CIDs in the Stereo-seq Chip T mask file and roughly estimate memory required to do `mapping`.
- **mapping:** Correspond *in situ* captured sequenced reads recorded in FASTQ[3,4] files by Stereo-seq with their spatial information. It also aligns reads to the reference genome and generates coordination sorted BAM files.
- **merge (optional):** Combine CID (same as barcodes) listed files with reads count from multiple runs of `mapping`. Only for an analysis that requires to combine multiple pairs of FASTQ.
- **count:** Read BAM files generated from `mapping` to perform gene annotation, de-duplication, and gene expression analysis on the aligned reads.
- **register:** Align microscopic tissue staining image with gene expression matrix file (GEF) generated from `count`.`register` is an optional pipeline when image fails QC or input image is absent.
- **imageTools:** Convert TIFF images from IPR, such as template-aligned stitched TIFF image, binarized tissue segmentation and cell segmentation images. Optional module when image fails QC or input image is absent.
- **tissueCut:** Identify tissue coverage area on the chip and extract gene expression matrix of the corresponding spatial location by taking inputs from both `count` and `register` or `count` pipeline alone.
- **spatialCluster:** Perform clustering analysis for spots (bin200) according to the gene expression matrix of the tissue coverage area generated from `tissueCut`.
- **cellCut:** Identify cell nuclei area on the staining image and extract gene expression matrix of the corresponding spatial location by taking inputs from both `count` and `register&imageTools` pipeline. Optional module when image fails QC or input image is absent.
- **cellCorrect:** Adjust cell coverage region based on the aligned cell nuclei segmentation image generated from `register` and `imageTools`. Then, output expression matrix of the adjusted cells in cell bin GEF and GEM formats.
- **cellCluster:** Perform clustering analysis for cell bins according to the gene expression matrix which is generated from `cellCorrect`. Optional module when image failed QC or input image is absent.
- **saturation:** Calculate sequencing saturation of tissue coverage area based on the file that was used for sampling data generated from `count`.
- **report:** Generate a JSON format statistical summary report that integrates the analysis result from each step, as well as an HTML web analysis report, and shows spatial expression distribution of genes, key statistical metrics, sequencing saturation plots, clustering analysis results. Depending on the image input state and register mode, HTML reports may or may not have cell bin statistical data and image processing key results.

Other handy functions:

- **Other applications of cellCut:** Manipulate GEF file.
- **checkGTF:** Check GTF/GFF file is prepared in the correct format, otherwise, re-format one that meets the compatibility requirements for `count`.

- **Other applications of imageTools:** 1) Merge two to three TIFF images in R-G-B order, which is useful for checking segmentation result. 2) Plot templates on the panoramic image to assist in the evaluation of stitching and registration result. 3) Write TIFF images into RPI format.
- **manualRegister:** Acquire manual registration operation parameters from **StereoMap visualization software** and run `manualRegister` to modify registration records in IPR. Users can turn on the "fine-tune" switch to let `manualRegister` perform an automatic adjustment to the manually operated result to make the registration more precise.
- **lasso:** Acquire one or multiple cell or spatial gene expression matrix subsets according to the GEOJSON file, which stores the coordinates data of manually delineated region(s) from **StereoMap visualization software**.
- **cellChunk:** Generate encoding pre-computed data used for **StereoMap** rendering.

1.2. System Requirements

SAW runs on Linux systems that meet the following minimum requirements:

8-core Intel or AMD processor (>24 cores recommended)
128GB RAM (>256GB recommended)
1TB free disk space or higher
64-bit CentOS/RedHat 7.8 or Ubuntu 20.04

To install and run SAW, please install one kind of the following softwares:

docker[5]**:** version 20.10.8 or higher
singularity[6]: version 3.8 or higher

1.3. Related Software

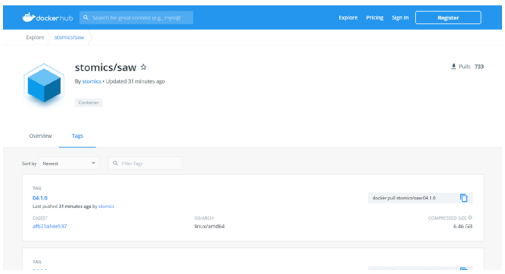
- **ImageQC: STOmics microscope ImageQC software** is a desktop application intended for assessing the quality of microscope images. The fluorescent image should pass **ImageQC** evaluation to ensure that it fulfills the requirements for SAW pipelines. SAW >= 5.0.0 requires **ImageQC** version >= 1.1.0 that provides IPR file for recording image processing data. The final version of **ImageQC** is v1.2.0 and will be no longer updated.
- **ImageStudio: ImageStudio** is a fully upgraded image process desktop application which integrates the entire **ImageQC** functionality. It contains four main image processing modules: image QC, manual stitching, manual tissue segmentation, and manual cell segmentation. The outputs of each module can be input into SAW for further analysis. SAW v7.0 recommends **ImageStudio** version >= v3.0.
- **StereoMap: StereoMap** is an HD visualization desktop application intended for displaying Stereo-seq analysis results. SAW outputs such as gene expression matrix GEF file, image RPI and IPR files, clustering results can be visualized via **StereoMap**. SAW v7.0 recommends **StereoMap** version >= v3.0.

1.4. SAW Docker Image Installation

SAW is delivered as a docker image that bundles all of its required software dependencies. You can pull the SAW docker image from Docker Hub to your local system and run analyses offline.

Please download the latest version of the software and use it with the corresponding **Stereo-seq Analysis Workflow Software Suite User Manual** version.

Docker Hub link: **https://hub.docker.com/r/stomics/saw/tags**



We support using Singularity and Docker to install and run the SAW.

Here we take **SAW version 6.1** as an example.

1.4.1. SAW Installation via Singularity

Please replace the yellow highlighted inputs with your own data path.

Step 1: Pull the SAW docker image (2 options):

```
$ singularity build SAW_v7.0.sif docker://stomics/saw:07.0.0  ## option 1
$ singularity build --sandbox SAW_v7.0/ docker://stomics/saw:07.0.0  ## option 2
```

For non-root users, especially who don't have enough space in the **/home/** directory, please try:

```
$ export SINGULARITY_CACHEDIR=/path/to/build
$ singularity build --sandbox SAW_v7.0/ docker://stomics/saw:07.0.0
```

Step 2: Run pipelines (3 options):

Please Note! All the requested paths need to be mounted before input. For example, it is necessary to bind directories that store input files (`/path/to/data`), reference genome (`/path/to/genomeDir`), and outputs (`/path/to/output`).

```
$ export SINGULARITY_BIND="/path/to/data,/path/to/genomeDir,/path/to/output"
```

Option 1: Run pipelines within the container from the host system.

```
$ /path/to/SAW_v7.0.sif <application>  ## option 1.1
$ singularity exec /path/to/SAW_v7.0.sif <application>  ## option 1.2
```

Option 2: Shell into the SAW container and interactively run bash commands. Run `exit` to exit the environment.

```
$ /path/to/SAW_v7.0.sif /bin/bash  ## option 2.1
Singularity>
Singularity> <shell-command>
Singularity> exit
$
$ singularity shell /path/to/SAW_v7.0.sif  ## option 2.2
Singularity>
Singularity> <shell-command>
Singularity> exit
$
$ singularity shell SAW_v7.0  ## option 2.3 for sandbox
Singularity>
Singularity> <shell-command>
Singularity> exit
$
```

Option 3: Use **"-B directory on the host-machine:directory in the container"** to mount a host directory into the container and execute the command in the container.

```
$ singularity shell -B /path/to/directory/on/the/host-machine:/path/to/directory/
mounted/in/the/container /path/to/SAW_v7.0.sif
Singularity>
Singularity> <shell-command>
Singularity> exit
$
```

* Please be noted that these two lines belong to the same line of command.

1.4.2. SAW Installation via Docker

Step 1: Pull the SAW docker image

```
$ docker pull stomics/saw:07.0.0
```

Step 2: Run pipelines

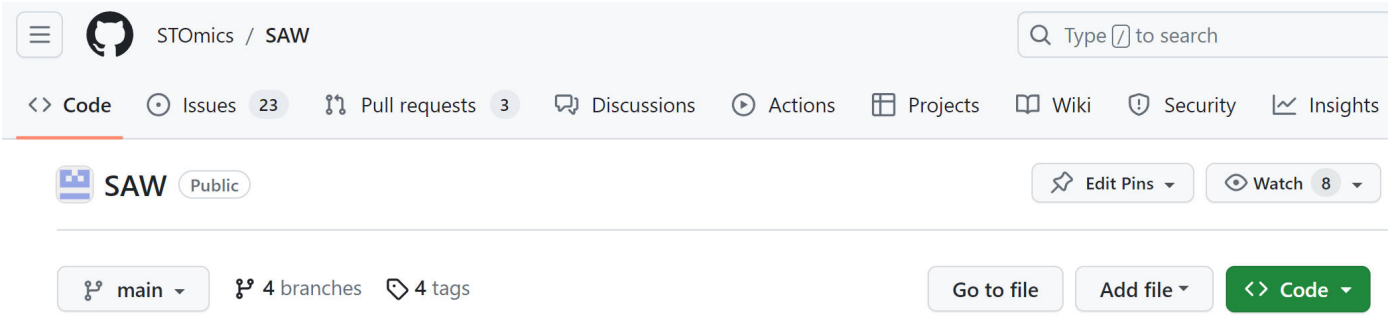
```
$ docker run -d -v /path/to/data:stomics/saw:07.0.0 /bin/sh -c "<shell-command>"
```

1.5. SAW Github

SAW GitHub: **https://github.com/STOmics/SAW**

Please visit GitHub for instruction regarding the **installation of singularity** and **indexing reference genome**. The page also provides **SAW shell script** examples for users.

Please Note! Please build your reference before running SAW analysis.



1.6. SAW Test Data

SAW test data can be downloaded from SAW GitHub page. Key outputs are shown in **Chapter 3 Test Data Demonstration**. The reference genome version for SAW testing is:

- genome-build: GRCm38.p6
- genome-version: GRCm38
- genome-date: 2012-01
- genome-build-accession: NCBI:GCA_000001635.8

1.7. SAW Output File Format

Please check **Stereo-seq File Format Manual** to get more information on SAW output files format.

CHAPTER 2
SAW PIPELINES & ARGUMENTS

2.1 splitMask

`splitMask` is designed for splitting Stereo-seq Chip T mask file according to the SE FASTQ CID. The split count is directly related to the number of split FASTQs while writing FASTQ out from sequencer.

Running **`splitMask`** requires the following files:

- Stereo-seq Chip T mask file **(.h5)**

Expected running time for 1 cm x 1 cm chip: ~5 min, Memory: ~3G

As input sequencing data, both Q40 FASTQ and Q4 FASTQ from Stereo-seq, which store the original sequencing data, are supported in SAW pipelines. Differences between them are mainly reflected in file size and how data and information are written in.
Q40 FASTQ, also called PE FASTQ, has a pair of read files, read 1 and read 2. Q4 FASTQ, also called SE FASTQ, is an output format with only one file, but saves file storage space.
More details in Stereo-seq File Format Manual.

An example of Q4 FASTQ file path usually displayed as below:

2.1.1 Arguments and Options

Table 2-1 `splitMask` Arguments and Options

Parameter index	Function
`[1]`	(Required) Stereo-seq Chip T mask file path (.h5).
`[2]`	(Required) Output directory stores split mask files.
`[3]`	(Required) Set the number of threads to be used.
`[4]`	(Optional) Split count. This count number needs to be set the same as Q4 FASTQ count. The options are the powers of 4, and the commonly used options are 16 and 64. If a library contains two barcodes sequenced in the same lane and the two barcodes were split when written out FASTQs, then the split count is 16; otherwise, if the barcodes were not split, then 64.
`[5]`	(Optional) CID position, commonly used option is "`2_25`". **`splitMask`** uses 24 bases (out of 25, 25 is the read length of CID) to split Stereo-seq Chip T mask file. This CID position parameter is a string that combines two index numbers with "_". The two numbers indicate the start and the end base index in the CID, and the bases in this range are used for splitting. For example, "`2_25`" means starting from the 2nd base to the 25th base used for splitting mask file.

2.1.2 Usage Example

Please Note! Replace **{SN}** with your Stereo-seq Chip T serial number (SN, e.g. SS200000135TL_D1 or A00135D1)

```
$ mkdir /path/to/output/00.splitmask
$ singularity exec SAW_v7.0.sif splitMask \
      /path/to/data/{SN}.barcodeToPos.h5 \  ## Mask path
      /path/to/output/00.splitmask \  ## output directory
      8 \  ## threads
      16 \  ## split count
      2_25  ## CID position
```

2.1.3 Outputs

The output files of **splitMask** are organized as below. The indices of the output files are corresponding to the split list index of the Q4 FASTQ files.

```
$ tree /path/to/output/00.splitmask
/path/to/output/00.splitmask
|-- 01.SN.barcodeToPos.bin
|-- 02.SN.barcodeToPos.bin
|-- 03.SN.barcodeToPos.bin
|-- 04.SN.barcodeToPos.bin
|-- 05.SN.barcodeToPos.bin
|-- 06.SN.barcodeToPos.bin
|-- 07.SN.barcodeToPos.bin
|-- 08.SN.barcodeToPos.bin
|-- 09.SN.barcodeToPos.bin
|-- 10.SN.barcodeToPos.bin
|-- 11.SN.barcodeToPos.bin
|-- 12.SN.barcodeToPos.bin
|-- 13.SN.barcodeToPos.bin
|-- 14.SN.barcodeToPos.bin
|-- 15.SN.barcodeToPos.bin
`-- 16.SN.barcodeToPos.bin

0 directories, 16 files
```

2.2 CIDCount

CIDCount is a program for computing the number of CIDs in the Stereo-seq Chip T mask file and roughly estimating memory used in **mapping**.

Running **CIDCount** requires the following files:

- Stereo-seq Chip T mask file **(.h5)**

Expected running time for 1 cm x 1 cm chip: ~30 s, Memory: ~0.7G

2.2.1 Arguments and Options

Table 2-2 `CIDCount` Arguments and Options

Parameter	Function
-i	(Required) Stereo-seq Chip T mask file path (.h5 or .bin).
-s	(Optional) A string of species name.
-g	(Required) Genome file size in GB.

2.2.2 Usage Example

For Q40 FASTQ input cases, run **CIDCount** as below:

```
$ singularity exec SAW_v7.0.sif CIDCount \
      -i /path/to/data/{SN}.barcodeToPos.h5 \  ## Stereo-seq Chip T mask file path
      -s {speciesName} \  ## species name
      -g {genomeSize}  ## genome file size in GB, can be acquired by “ls -l
--block-size=GB ${Genome file of the species after STAR indexing}”
```

For Q4 FASTQ input cases that require to run **splitMask** first, users may run **CIDCount** only once because the results for each individual small mask are close to each other. For chip size larger than 1 cm x 1 cm, we would recommend to run **CIDCount** for each CID class/small mask.

Please Note! **{index}** needs to be replaced by the real index number of the split mask file.

```
$ singularity exec SAW_v7.0.sif CIDCount \
      -i /path/to/output/00.splitmask/{index}.{SN}.barcodeToPos.bin \  ## Ste-
reo-seq Chip T mask file path
      -s {speciesName} \  ## species name
      -g {genomeSize}  ## genome file size in GB, can be acquired by “ls -l
--block-size=GB ${Genome file of the species after STAR indexing}”
```

2.2.3 Outputs

The output of **CIDCount** is shown as below,

```
$ singularity exec SAW_v7.0.sif CIDCount -i SN.barcodeToPos.h5 -s mouse -g 3
645784920  ## CID count
62  ## estimated memory for mapping
```

2.3 mapping

Each Stereo-seq sequenced read contains a CID sequence which is used as a key to spatially map the read back to its original location on the tissue section. **`mapping`** pipeline matches CID of the original sequencing reads stored in the FASTQ file with the records of CID-coordinates key-value pairs saved in the Stereo-seq Chip T mask file (allow 1 mismatch). Coordinate information for reads that CID could be paired with, based on the records of the mask file, will be added. Coordinate information for reads that CID could be paired with will be added based on the records of the mask file. Reads that get the coordinate annotations are Valid CID mRNA Reads (**Valid CID Reads**). After discarding reads with unqualified MID reads which do not satisfy further analysis, filtering reads with adapter, and filtering short reads (read length less than 30 after trimming consecutive A bases), the filtered **Valid CID Reads** are the **Clean Reads**. **`mapping`** pipeline maps **Clean Reads** to the reference genome, and output sorted BAM[7] format alignments and summary report.

If concerned about rRNA count and filtration, necessary modification to the reference genome should be completed beforehand, mainly including 2 steps: 1) add rRNA information into the FASTA file, with the suffix '_rRNA' on the chromosome column, like '1_rRNA', to distinguish them from original reference; 2) build index using modified reference genome.

Running **`mapping`** requires the following files:

- Stereo-seq sequenced reads FASTQ files **(.fq.gz)**
- Stereo-seq Chip T mask file **(.h5)**
- Indexed reference genome
- bcPara file **(.bcPara)**, please check the content of **Table 2-4**

Expected running time for ~1G reads: ~4 h, Memory: ~67G

Notice to rebuild reference index, with the same command input as before. From SAW V6.1 onwards, SAW reconstructed the construction of reference index and mapping algorithms for more efficient computational performance and reduced time cost.

2.3.1 Arguments and Options

As **`mapping`** encapsulate STAR[8] function, it accepts additional options beyond those shown in the table below.

Table 2-3 `mapping` Arguments and Options

Parameter	Function
`--outSAMattributes spatial`	Set to turn on spatial BAM file format mode.
`--outSAMtype BAM SortedByCoordinate`	(STAR option) Set output BAM file sorted by coordinate.
`--genomeDir`	(STAR option) Path to the directory where the genome indices are stored.
`--runThreadN`	(STAR option; defaults to 1) Set the number of threads to be used. Usually set to 8 or higher.
`--outFileNamePrefix`	(STAR option) Custom output file prefix.
`--sysShell /bin/bash`	(STAR option) Path to the shell binary.
`--stParaFile`	(Required) Create a parameter file that defines CID mapping options. Options are specified in Table 2-4.
`--readNameSeparator \" \"`	(STAR option) Character(s) separating the part of the read names that will be trimmed in output.

Parameter	Function
`--limitBAMsortRAM`	(STAR option) Maximum available RAM (bytes) for sorting BAM.
`--limitOutSJcollapsed`	(STAR option) Max number of collapsed junctions.
`--limitIObufferSize`	(STAR option) Max available buffers size (bytes) for input/output, per thread.
`--outSAMmultNmax`	(STAR option; defaults to -1) Max number of alignments that will be written in output BAM file for a read. Recommend setting to 1 to reduce disk space storage.

Table 2-4 `mapping --stParaFile` Arguments and Options

Parameter	Function
`in`	(Required) Path to the Stereo-seq Chip T mask file (PE) or split mask file (SE).
`in1`	(Required) Path to the FASTQ file. For Q40 sequences specify the path to the FASTQ file of read1 here. For Q4 sequences, input FASTQ file with the same split index as the mask file. A more elegant way to input a long list of Q4 FASTQs is to organize them in a `FQ_{index}.list` file. Please check **2.3.2 Usage Example Q4 scenario 2** to get more information.
`in2`	(Optional) Path to the FASTQ file of read2. Only valid for Q40 sequencing.
`barcodeReadsCount`	(Required) Mapped CID list file with read counts for each CID. This is an output file in `mapping`
`barcodeStart`	(Required; defaults to 0) CID start position. Set to 0.
`barcodeLen`	(Required; defaults to 25) CID length. Set to 25 for Q40, 24 for Q4.
`umiStart`	(Required; defaults to 25) MID start position.
`umiLen`	(Required; defaults to 10) MID length.
`mismatch`	(Required; defaults to 0) Max mismatch tolerant. Usually set to 1 in `mapping`.
`bcNum`	(Required) CID count in mask. Please check **2.2 CIDCount** for more information.
`polyAnum`	(Optional) Number of consecutive A in the read that will be trimmed. Recommend to set this value to 15.
`mismatchInPolyA`	(Optional) Number of mismatch bases in searching poly A. Recommend to set this value to 2
`rRNAremove`	(Optional; default to disable) Remove rRNA reads. Note that the precondition of the switch is the reference genome combined with rRNA information. If not, this filtering function will not work anyway. Using the reference genome with the addition of rRNA, reads related to rRNA are not filtered by default, namely the switch is off. And the number and the ratio of rRNA reads in the statistical output file are both 0. When the switch is on, as long as the query reads are mapped to rRNA reference, they will be removed. rRNA count and ratio are recorded in the output file.
`validCidFq`	(Optional; default to disable) Output **Valid CID Reads** in FASTQ format after CID mapping. Similar to Q4 FASTQ format but with different first row of each read, for instance, "@V350044321L1C001R0020993658|Cx:i:10413|Cy:i:7737 D3450E0D391E EC7FF", where Cx and Cy represent the decoded coordinates and are added after readID.
`outUnMappedFq`	(Optional; default to disable) Output un-mapped reads in FASTQ format after mapping **Clean Reads** to the reference genome. Similar to Q4 FASTQ format but with different first row of each read, for instance, "@V350044321L1C001R0020993658|Cx:i:10413|Cy:i:7737 D3450E0D391E EC7FF", where Cx and Cy represent the decoded coordinates and are added after readID.
`bcMappingRateCutoff`	(Optional; default to disable) Set the minimum CID mapping rate, like 'bcMappingRateCutoff=0.1' . If the actual CID mapping rate less than this ratio, abort `mapping`.

2.3.2 Usage Example

Two scenarios for preparing **mapping** --**stParaFile** input file **{lane}.bcPara** with Q40 FASTQ input:

Please Note! Replace {SN} with your Stereo-seq Chip T serial number (SN, e.g. SS200000135TL_D1 or A00135D1), and {lane} with the FASTQ lane name prefix (e.g. E100026571_L01)
The same applies to all examples.

Q40 scenario 1: Prepare **{lane}.bcPara** file for Q40 one pair FASTQ as **mapping** input:

```
$ mkdir /path/to/output/01.mapping
$ vim /path/to/output/01.mapping/{lane}.bcPara
in=/path/to/data/{SN}.barcodeToPos.h5
in1=/path/to/data/{lane}_read_1.fq.gz
in2=/path/to/data/{lane}_read_2.fq.gz
barcodeReadsCount=/path/to/ouptut/01.mapping/{lane}.barcodeReadsCount.txt
barcodeStart=0
barcodeLen=25
umiStart=25
umiLen=10
mismatch=1
bcNum=645784920  ## Input the first line from output of CIDCount
polyAnum=15
mismatchInPolyA=2
```

Q40 scenario 2: Prepare **{lane}.bcPara** file for Q40 multiple pair of FASTQ as **mapping** input:

```
$ mkdir /path/to/multi_lane_output/01.mapping
$ vim /path/to/multi_lane_output/01.mapping/{lane}.bcPara
in=/path/to/data/{SN}.barcodeToPos.h5
in1=/path/to/data/{lane}_read_1.fq.gz
in2=/path/to/data/{lane}_read_2.fq.gz
barcodeReadsCount=/path/to/multi_lane_output/01.mapping/{lane}.barcodeReadsCount.
txt
barcodeStart=0
barcodeLen=25
umiStart=25
umiLen=10
mismatch=1
bcNum=645784920  ## Input the first line from output of CIDCount
polyAnum=15
mismatchInPolyA=2
```

Run **mapping** for Q40 input:

```
$ singularity exec SAW_v7.0.sif mapping \
                --outSAMattributes spatial \
                --outSAMtype BAM SortedByCoordinate \
                --genomeDir /path/to/genomeDir \
                --runThreadN 8 \
                --outFileNamePrefix /path/to/output/01.mapping/{lane}. \
                --sysShell /bin/bash \
                --stParaFile /path/to/output/01.mapping/{lane}.bcPara \
                --readNameSeparator \" \" \
                --limitBAMsortRAM 38582880124 \
                --limitOutSJcollapsed 10000000 \
                --limitIObufferSize=280000000 \
                --outBAMsortingBinsN 50 \
                > /path/to/output/01.mapping/{lane}.run.log
```

Two scenarios for preparing **mapping** --**stParaFile** input file **{index}.bcPara** with Q4 FASTQ input:

Q4 scenario 1: Prepare **{index}.bcPara** file for Q4 FASTQ that contains one barcode in library or did not split barcode when writing FASTQ:

Please Note! {index} need to be replaced by the index of the split mask file which is corresponding to the index of the Q4 FASTQ file.

```
$ mkdir /path/to/output/01.mapping
$ vim /path/to/output/01.mapping/{index}.bcPara
in=/path/to/output/00.splitmask/{index}.{SN}.barcodeToPos.bin  ## split mask file
in1=/path/to/data/{lane}_{index}.fq.gz  ## {index}th FASTQ file in {lane}
barcodeReadsCount=/path/to/ouptut/01.mapping/{index}.barcodeReadsCount.txt
barcodeStart=0
barcodeLen=24
umiStart=25
umiLen=10
mismatch=1
bcNum=38284877  ## Input the first line from output of CIDCount
polyAnum=15
mismatchInPolyA=2
```

Q4 scenario 2: Prepare **`{idx}.bcPara`** file for Q4 FASTQ that has multiple barcodes in the library and split when writing FASTQ:

Please Note! **`{index}`** needs to be replaced by the real index number of the split mask file and **`{idx}`** needs to be replaced by the index number of the input FASTQ file. For example, **`{index}`** and **`{idx}`** of the first barcode group are"01-16" and "01-16", **`{index}`** and **`{idx}`** of the second barcode group are "01-16" and "17-32".

```
$ mkdir /path/to/output/01.mapping
$ vim /path/to/output/01.mapping/{idx}.bcPara
in=/path/to/output/00.splitmask/{index}.{SN}.barcodeToPos.bin  ## split mask file
in1=/path/to/data/{lane}_{index}.fq.gz  ## {index}th FASTQ file in {lane}. FASTQ files for different barcode usually stores in different directory, so /path/to/data/ might change to /path/to/data/{barcode_n}
barcodeReadsCount=/path/to/ouptut/01.mapping/{idx}.barcodeReadsCount.txt
barcodeStart=0
barcodeLen=24
umiStart=25
umiLen=10
mismatch=1
bcNum=38284877 ## Input the first line from output of CIDCount
polyAnum=15
mismatchInPolyA=2
```

In case there are too many FASTQ files that need to be processed, an easier way is to organize them into a **FQ_{index}.list** file. The requirement for preparing a **FQ_{index}.list** file is to gather all the FASTQs with the same index as the split mask together, since these files are all split by the same logic.

Get the list input conveniently like:

```
$ sh manage.sh <FASTQDirList> <outdir> <SN>

FASTQDirList = <Q4 FASTQ directory list which record directory paths line byline>
outDir = <directory to output FQ.list files>
SN = <Stereo-seq chip serial number>
```

https://github.com/STOmics/SAW/blob/main/Scripts/Get_FASTQ_list.sh

An example to display FQ.list file :

```
$ cat SN_Q4_fastq_16.list  ## a FASTQ list file gathers all the FASTQs whose index is 16
/path/to/data/lane_1_16.fq.gz
/path/to/data/lane_2_16.fq.gz
```

Then input the path of **FQ_{index}.list** file for **in1** parameter in the **{idx}.bcPara** file. The **{index}** of **FQ_{index}.list** and the **{index}** of split mask file have to be the same.

```
$ mkdir /path/to/output/01.mapping
$ vim /path/to/output/01.mapping/{index}.bcPara
in=/path/to/output/00.splitmask/{index}.{SN}.barcodeToPos.bin  ## split mask file
in1=/path/to/output/{SN}_Q4_fastq_{index}.list
barcodeReadsCount=/path/to/output/01.mapping/{idx}.barcodeReadsCount.txt
barcodeStart=0
barcodeLen=24
umiStart=25
umiLen=10
mismatch=1
bcNum=38284877 ## Input the first line from output of CIDCount
polyAnum=15
mismatchInPolyA=2
```

Run **mapping** pipeline

```
$ singularity exec SAW_v7.0.sif mapping \
                --outSAMattributes spatial \
                --outSAMtype BAM SortedByCoordinate \
                --genomeDir /path/to/genomeDir \
                --runThreadN 8 \
                --outFileNamePrefix /path/to/output/01.mapping/{index}. \  ## {index} or {idx} depending on the scenario
                --sysShell /bin/bash \
                --stParaFile /path/to/output/01.mapping/{index}.bcPara \  ## {index} or {idx} depending on the scenario
                --readNameSeparator \" \" \
                --limitBAMsortRAM 38582880124 \
                --limitOutSJcollapsed 10000000 \
                --limitIObufferSize=280000000 \
                --outBAMsortingBinsN 50 \
                > /path/to/output/01.mapping/{index}.run.log  ## {index} or {idx} depending on the scenario
```

2.3.3 Outputs

Q40 scenario 1 output files are organized as below:

```
$ tree /path/to/output/01.mapping/
/path/to/output/01.mapping/
├── lane.Aligned.sortedByCoord.out.bam
├── lane.barcodeReadsCount.txt
├── lane.bcPara
├── lane.CIDMap.stat
├── lane.Log.final.out
├── lane.Log.out
├── lane.Log.progress.out
├── lane.run.log
└── lane.SJ.out.tab
```

Q40 scenario 2 output files are organized as below (Here shows an example of 2 pairs of FASTQ):

If one sample has multiple FASTQ files, you need to run `mapping` for each FASTQ pair.

```
$ tree /path/to/multi_lane_output/01.mapping
/path/to/multi_lane_output/01.mapping
├── lane1.Aligned.sortedByCoord.out.bam
├── lane1.barcodeReadsCount.txt
├── lane1.bcPara
├── lane1.CIDMap.stat
├── lane1.Log.final.out
├── lane1.Log.out
├── lane1.Log.progress.out
├── lane1.run.log
├── lane1.SJ.out.tab
├── lane2.Aligned.sortedByCoord.out.bam
├── lane2.barcodeReadsCount.txt
├── lane2.bcPara
├── lane2.CIDMap.stat
├── lane2.Log.final.out
├── lane2.Log.out
├── lane2.Log.progress.out
├── lane2.run.log
└── lane2.SJ.out.tab
```

Q4 scenario 1 (one barcode in library or did not split barcode when writing FASTQ) output files are organized as below:

```
$ tree /path/to/output/01.mapping
/path/to/output/01.mapping
├── 01.Aligned.sortedByCoord.out.bam
├── 01.barcodeReadsCount.txt
├── 01.bcPara
├── 01.CIDMap.stat
├── 01.Log.final.out
├── 01.Log.out
├── 01.Log.progress.out
├── 01.run.log
├── 01.SJ.out.tab
...
├── 16.Aligned.sortedByCoord.out.bam
├── 16.barcodeReadsCount.txt
├── 16.bcPara
├── 16.CIDMap.stat
├── 16.Log.final.out
├── 16.Log.out
├── 16.Log.progress.out
├── 16.run.log
└── 16.SJ.out.tab
```

Q4 scenario 2 (multiple barcodes in the library and split when writing FASTQ) output files are organized as below (Here shows an example of 2 barcodes):

```
$ tree /path/to/output/01.mapping
/path/to/output/01.mapping
/path/to/output/01.mapping
├── 01.Aligned.sortedByCoord.out.bam
├── 01.barcodeReadsCount.txt
├── 01.bcPara
├── 01.CIDMap.stat
├── 01.Log.final.out
├── 01.Log.out
├── 01.Log.progress.out
├── 01.run.log
├── 01.SJ.out.tab
...
├── 128.Aligned.sortedByCoord.out.bam
├── 128.barcodeReadsCount.txt
├── 128.bcPara
├── 128.CIDMap.stat
├── 128.Log.final.out
├── 128.Log.out
├── 128.Log.progress.out
├── 128.run.log
└── 128.SJ.out.tab
```

2.4 merge (optional)

SAW **merge** pipeline is used to combine the results of **mapping**.

Running **merge** requires the following file:

- **mapping** output mapped CID list files **(.txt)**

Expected running time for ~1G reads 2 lanes: ~5 min, Memory: ~5G

2.4.1 Arguments and Options

Table 2-5 merge Arguments and Options

Parameter	Function
[1]	(Required) Path to the Stereo-seq Chip T mask file.
[2]	(Required) Mapped CID list files with read counts for each CID.
[3]	(Required) Mapped CID list file which merges all input files.

2.4.2 Usage Example

```
$ mkdir /path/to/multi_lane_output/02.merge
$ singularity exec SAW_v7.0.sif merge \
      /path/to/data/{SN}.barcodeToPos.h5 \
      /path/to/multi_lane_output/01.mapping/{lane1}.barcodeReadsCount.txt,/path/to/
multi_lane_output/01.mapping/{lane2}.barcodeReadsCount.txt \  ## change {lane} to
{index} or {idx} for Q4 and change /path/to/multi_lane_output/ to /path/to/output/
for SE single lane scenario
      /path/to/multi_lane_output/02.merge/{SN}.barcodeReadsCount.txt
```

* Please be noted that we use the backward slash "\" to indicate the end of a line in a command that spans multiple lines.

2.4.3 Ouputs

The output file of **merge** has been organized as below:

```
$ tree /path/to/multi_lane_output/02.merge
/path/to/multi_lane_output/02.merge
└── {SN}.merge.barcodeReadsCount.txt
```

2.5 count

SAW **count** is an efficient general-purpose read annotation pipeline that label reads with their overlapped genomic features and outputs statistical information for the overall summarization result. Through quantification of annotated reads, **count** generates spatial gene expression data after de-duplicating reads according to CID, gene ID, and MID information. Usually, SAW **count** is run on the **Uniquely Mapped Reads** filtered from **mapping** output based on the reference genome annotation records. Starting from SAW v5.1.3, **count** allows the utilization of some Multi-Mapped Reads in quantification (**--multi_map**). The gene expression level in SAW pipeline is the summation of both intron and exon MID count. To support downstream analysis that might be required of different genomic features, **count** also outputs exon MID count separately.

Running **count** requires the following files:

- **mapping** output BAM file **(.bam)**
- Reference genome annotation GFF/GTF[9,10] file **(.gff / .gtf)**

Expected running time for ~1G reads: 0.5 h, Memory: ~45 G

2.5.1 Arguments and Options

Table 2-6 count Arguments and Options

Parameter	Function
-i	(Required) **mapping** output BAM file. Separate multiple files by comma.
-o	(Required) Set the **count** output BAM file name.
-a	(Required) Gene annotation GFF/GTF file.
-s	(Required) Set the **count** output statistical summary report file name.
-e	(Required) Set the **count** output gene expression file name.
--umi_len	(Required; defaults to 10) MID length.
--sn	(Required) Stereo-seq Chip T serial number (SN).
-c	(Optional; defaults to detected) CPU core number to use. Minimum value is 8, recommended value is 24.
--save_lq	(Optional; defaults to false) Save low-quality reads if set.
--save_dup	(Optional; defaults to false) Save duplicate reads if set.
--umi_on	(Optional; defaults to false) Correct MID if set.
--sat_file	(Optional; defaults to None) Set the saturation sampling file name which is prepared for sequencing saturation (requires **--umi_on**).
--multi_map	(Optional; defaults to disable) Set to enable multi-mapped reads correction. This correction consists of two logics. 1) The first logic is the correction of multi-gene reads which is a read mapped uniquely to a genomic region where multiple genes overlap. In this scenario, the read has to overlap with a genomic locus greater than 50% of its read length. If the genomic feature (exon, intron, or intergenic) of all the mapped genes includes exon and intron, then select the alignment record mapped to exon in preference to intron. If more than one gene locus belongs to the same genomic feature type, then pick the one that has the longest overlap. Otherwise, label the read to “intergenic.” 2) The second logic is to select and correct one of the multi-mapped reads and add the count to gene expression matrix. The first step is to group reads by QNAME. Select reads in the group mapped to exon in preference to those mapped to intron. Then correct the longest overlapped reads (at least overlapped greater than 50% of its read length) in the group to unique read and correct its MAPQ to 255. Set the MAPQ of the remaining reads to 0.

2.5.2 Usage Example

```
$ mkdir -p /path/to/output/03.count
$ geneExp=/path/to/output/03.count/{SN}.raw.gef
$ saturationSamplingFile=/path/to/output/03.count/{SN}_raw_barcode_gene_exp.txt
$ singularity exec SAW_v6.1.sif count \
        -i /path/to/output/01.mapping/{lane}.Aligned.sortedByCoord.out.bam \
        -o /path/to/output/03.count/{SN}.Aligned.sortedByCoord.out.merge.q10.dedup.
target.bam \
        -a /path/to/reference/genes.gtf \
        -s /path/to/output/03.count/{SN}.Aligned.sortedByCoord.out.merge.q10.dedup.
target.bam.summary.stat \
        -e ${geneExp} \
        --umi_len 10 \
        --sat_file ${saturationSamplingFile} \
        --sn {SN} \
        --umi_on \
        --save_lq \
        --save_dup \
        -c 24
```

* Please be noted that we use the backward slash "\" to indicate the end of a line in a command that spans multiple lines.

For more than one pair of FASTQ files (Here shows an example of 2 pairs of FASTQ):

```
$ mkdir -p /path/to/multi_lane_output/03.count  ## change /path/to/multi_lane_out-
put/ to /path/to/output/ for SE single lane scenario
$ geneExp=/path/to/multi_lane_output/03.count/{SN}.raw.gef
$ saturationSamplingFile=/path/to/multi_lane_output/03.count/{SN}_raw_barcode_gene_
exp.txt
$ singularity exec SAW_v7.0.sif count \
        -i /path/to/multi_lane_output/01.mapping/{lane1}.Aligned.sortedByCoord.out.
bam,/path/to/ multi_lane_output/01.mapping/{lane2}.Aligned.sortedByCoord.out.bam \
## change {lane} to {index} or {idx} for SE
        -o /path/to/multi_lane_output/03.count/{SN}.Aligned.sortedByCoord.out.
merge.q10.dedup.target.bam \
        -a /path/to/reference/genes.gtf \
        -s /path/to/multi_lane_output/03.count/{SN}.Aligned.sortedByCoord.out.
merge.q10.dedup.target.bam.summary.stat \
        -e ${geneExp} \
        --umi_len 10 \
        --sat_file ${saturationSamplingFile} \
        --sn {SN} \
        --umi_on \
        --save_lq \
        --save_dup \
        -c 24
```

* Please be noted that we use the backward slash "\" to indicate the end of a line in a command that spans multiple lines.

For dealing with multi-mapped reads:

```
$ singularity exec SAW_v7.0.sif count \
        -i /path/to/output/01.mapping/{lane}.Aligned.sortedByCoord.out.bam \
        -o /path/to/output/03.count/{SN}.Aligned.sortedByCoord.out.merge.q10.dedup.
target.bam \
        -a /path/to/reference/genes.gtf \
        -s /path/to/output/03.count/{SN}.Aligned.sortedByCoord.out.merge.q10.dedup.
target.
bam.summary.stat \
        -e ${geneExp} \
        --umi_len 10 \
        --sat_file ${saturationSamplingFile} \
        --sn {SN} \
        --umi_on \
        --save_lq \
        --save_dup \
        -c 24 \
        --multi_map
```

2.5.3 Ouputs

The **count** output files are organized as below:

```
$ tree /path/to/output/03.count
/path/to/output/03.count
├── SN.Aligned.sortedByCoord.out.merge.q10.dedup.target.bam
├── SN.Aligned.sortedByCoord.out.merge.q10.dedup.target.bam.csi
├── SN.Aligned.sortedByCoord.out.merge.q10.dedup.target.bam.summary.stat
├── SN_raw_barcode_gene_exp.txt
└── SN.raw.gef
```

When you run `mapping` using reference genome with rRNA information and the switch of `rRNAremove` is on, `PASS FILTER` item in `*.bam.summary.stat` file will not contain the number of the reads mapped to rRNA reference.

2.6 register

SAW **`register`** pipeline aligns the microscopic tissue staining images with the plot of gene expression matrix generated by **`count`** based on the track lines on the chip while establishing the mapping relationship between images and spatial gene expression distribution. Staining images include:

- DAPI/ssDNA image for cell nucleus;
- IF (immunofluorescence) image for protein.
- H&E (hematoxylin and eosin) image for cell nuclei, extracellular matrix and cytoplasm.

SAW **`register`** includes four main modules, stitching, tissue segmentation, cell segmentation, and registration. Stitching combines microscopic images with overlapping sections to create a panoramic image (skip stitching if the input image is already a panoramic image). Tissue and cell segmentation modules detect and mask out tissue and cell coverage region, respectively. Registration module aligns the stitched image and the expression matrix, and at the same time registered masks from segmentation modules with the gene expression matrix using the same parameters. When it comes to the scenario of DAPI and mIF (multiple immunofluorescence images), this module outputs the registered DAPI and mIF image files. Tissue segmentation of IF image adopts the method of threshold segmentation, while cell segmentation of it is the intersection result of IF tissue segmentation, DAPI tissue segmentation and DAPI cell segmentation.

In addition, stitching and segmentation modules can be run separately with registration.

QC-failed data which is manually processed are supported currently. The tissue or cell segmentation image will adopt the manually processed results from ImageStudio (Both QC failure and success are accepted). More to the manual tutorial on GitHub.

Running **`register`** requires the following files:

- **`count`** output gene expression matrix file **(.raw.gef)**
- The microscopic tissue staining image file **(.tar.gz)** processed by **ImageStudio.** **`register`** is compatible with QC outputs from **ImageStudio** v3.
- **ImageStudio** Image process record file **(.ipr)**

Expected running time for 1 cm x 1 cm Stereo-seq Chip T image: ~7.5 h, Memory: ~20 G

Cell segmentation is the most time-consuming part. You may use -w False to skip cell segmentation, but still perform stitching, tissue segmentation and registration.

2.6.1 Arguments and Options

Table 2-7 **`register`** Arguments and Options

Parameter	Function
`-i`	(Required) ImageQC/ImageStudio processed staining image TAR.GZ file or image pre-processed output directory.
`-c`	(Required) ImageQC/ImageStudio IPR (image process record) file. IPR is designed to efficiently hold images and process records generated from each image processing step along with metadata in various data types. Please check **Stereo-seq File Format Manual** to get more information on the IPR file.
`-v`	(Optional. Depends on `-i`) **`count`** output gene expression matrix GEF file.
`-o`	(Required) Path to the directory where to store the **`register`** outputs.

Parameter	Function
-w	(Required, bool) Whether to do cell segmentation.
-P	(Optional, str=None) Customized IF thresholds and separated by comma and correspond to -g, e.g "-p 70,70".
-g	(Optional, str=None) IF group names and separated by comma, e.g "-g 355_ID,648_IF".
--gpu	(Optional, str=-1) Set GPU ID. eg: "--gpu 0". Default using CPU by "--gpu -1".
--core	(Optional, int=os.cpu_count()) Set CPU core number.

Preparation of GPU Usage:

pip install onnxruntime-gpu==1.15.1
export LD_LIBRARY_PATH=/usr/local/cuda/lib64:$LD_LIBRARY_PATH

2.6.2 Usage Example

Scenario 1: Process images from ImageQC/ImageStudio output data and gene expression matrix. Perform stitching, tissue segmentation, cell segmentation, and registration with gene expression matrix.

```
$ image=/path/to/data/image
$ image4register=$(find ${image} -maxdepth 1 -name {SN}*.tar.gz | head -1)
$ imageQC=$(find ${image} -maxdepth 1 -name {SN}*.ipr | head -1)
$ mkdir -p /path/to/output/04.register
$ singularity exec SAW_v7.0.sif register \
                    -i ${image4register} \
                    -c ${imageQC} \
                    -v /path/to/output/03.count/{SN}.raw.gef \
                    -o /path/to/output/04.register
                    -w True ## do cell segmentation
```

Scenario 2: Process images from ImageQC/ImageStudio output data. Perform stitching, tissue segmentation, and cell segmentation without doing registration with the gene expression matrix.

```
$ image=/path/to/data/image
$ image4register=$(find ${image} -maxdepth 1 -name {SN}*.tar.gz | head -1)
$ imageQC=$(find ${image} -maxdepth 1 -name {SN}*.ipr | head -1)

$ mkdir -p /path/to/output/04.register
$ singularity exec SAW_v7.0.sif register \
    -i ${image4register} \
    -c ${imageQC} \
    -o /path/to/output/04.register
    -w True ## do cell segmentation
```

Scenario 3: Process images from scenario 2. `register` processed images with gene expression matrix.

```
$ imageQC=$(find /path/to/output/04.register -maxdepth 1 -name {SN}*.ipr | head -1)

$ mkdir -p /path/to/output/04.register
$ singularity exec SAW_v7.0.sif register \
    -i /path/to/output/04.register \  ## -i input scenario 2 output directory
    -c ${imageQC} \  ## scenario 2 output IPR
    -v /path/to/output/03.count/{SN}.raw.gef \
    -o /path/to/output/04.register
    -w False ## cell segmentation has been done in Scenario 2
```

2.6.3 Outputs

`register` output files of scenario 1 and scenario 3 (if `-i` and `-o` are identical) are organized as below:

```
$ tree /path/to/output/04.register
/path/to/output/04.register
...  ## skip setting files, logs and image folder
├── <stainType>_fov_stitched_transformed.tif
└── SN_<chipType>_date_time_version.ipr
```

`register` output files of scenario 2:

```
$ tree /path/to/output/04.register
/path/to/output/04.register
...  ## skip setting files, logs and image folder
├── <stainType>_fov_stitched_transformed.tif
└── SN_<chipType>_date_time_version.ipr
```

`register` output files of scenario 3 if `-i` and `-o` are two different paths:

```
$ tree /path/to/output/04.register
/path/to/output/04.register
...  ## skip logs
└── SN_<chipType>_date_time_version.ipr
```

2.7 imageTools

SAW **`imageTools`** is a handy toolkit designed to play with image data in SAW. **`imageTools ipr2img`** (or **`ipr2img`**, available from SAW ST >= v5.0.0) is required in SAW core pipeline "decode" images from the processed IPR file. SAW **`imageTools ipr2img`** outputs TIFF images from IPR such as pre-registered ssDNA stitched image, tissue segmentation and cell segmentation mask TIFFs, as well as registered three types of images.

Running **`imageTools ipr2img`** requires the following files:

- ImageQC/ImageStudio processed microscopic tissue staining image file **(.tar.gz)**
- **`register`** processed Image process record file **(.ipr, compatible with v0.0.1 and v0.1.0)**

Expected running time for 1 cm x 1 cm Stereo-seq Chip T image: ~5 min, Memory: ~10 G

2.7.1 Arguments and Options

Table 2-8 `imageTools ipr2img` Arguments and Options

Command	Parameter	Function
`ipr2img`	**`-i`**	(Required) ImageQC/ImageStudio processed staining image TAR.GZ file.
	`-c`	(Required) IPR (image process record) file. IPR is designed to efficiently hold image information generated from each processing step and datasets in various compound types along with metadata. Please check **Stereo-seq File Format Manual** to get more information on IPR file.
	`-d`	(Required) Segmentation module names. Convert segmentation mask TIFF from IPR for the selected modules. Valid options: `tissue` and `cell`. Separate modules by space.
	`-r`	(Required; default to True) Whether output registered images or pre-registered images. “Pre-registered” state stands for the images that have been stitched to a single panoramic image and transformed to have the same scale with the gene expression matrix, but still need to be flipped, 90°rotated, or translated. The options are `True` for output registered images and `False` for output pre-registered images.
	`-o`	(Required) Path to the directory where to store the `ipr2img` outputs.

Please check **2.14.4 Other applications of imageTools** to learn more about **`imageTools`**.

2.7.2 Usage Example

```
$ image=/path/to/data/image
$ image4register=$(find ${image} -maxdepth 1 -name {SN}*.tar.gz | head -1)
$ imageIPR=$(find /path/to/output/04.register -maxdepth 1 -name {SN}*.ipr | head -1)
## has to be a processed IPR

$ singularity exec SAW_v7.0.sif imageTools ipr2img \
      -i ${image4register} \
      -c ${imageIPR} \
      -d tissue cell \  ## output both tissue and cell segmentation mask TIFF
      -r True \
      -o /path/to/output/04.register
```

2.7.3 Outputs

imageTools ipr2img output files (if the parent directory path of -c IPR and -o are identical) are organized as below:

```
$ tree /path/to/output/04.register
/path/to/output/04.register
...  ## skip setting files, logs and image folder
├── <stainType>_fov_stitched_transformed.tif
├── <stainType>_matrix_template.txt
├── <stainType>_SN_mask.tif
├── <stainType>_SN_regist.tif
├── <stainType>_SN_tissue_cut.tif
├── <stainType>_transform_template.txt
├── fov_stitched_transformed.rpi
├── SN.rpi
└── SN_<chipType>_date_time_version.ipr
```

imageTools ipr2img output files (if the parent directory path of -c IPR and -o are two different paths) are organized as below:

```
$ tree /path/to/output/04.register_tmp
/path/to/output/04.register_tmp
...  ## skip logs and image folder
├── <stainType>_fov_stitched_transformed.tif
├── <stainType>_matrix_template.txt
├── <stainType>_SN_mask.tif
├── <stainType>_SN_regist.tif
├── <stainType>_SN_tissue_cut.tif
├── <stainType>_transform_template.txt
└── SN.rpi
```

2.8 tissueCut

SAW **tissueCut** pipeline can delineate and extract the tissue coverage area based on the aligned image generated from **register** and **imageTools** or from the plot of gene expression matrix (if microscopic tissue staining images are not available). **tissueCut** outputs expression data in GEF format.

If the output of tissueCut doesn't match the morphology of the tissue, users could handle this issue with the help of ImageStudio, StereoMap or Stereopy[11]. If the tissueCut was run with an image, users could manually redo tissue segmentation for the image via ImageStudio and run register, imageTools and tissueCut again. Otherwise, manually delineate tissue coverage region from the gene expression distribution plot via StereoMap and then put the result into the SAW-lasso pipeline (2.15.6 lasso), or do lasso selection and expression matrix extraction via Stereopy (Stereopy->Tutorials->MatrixExport).

Running **tissueCut** requires the following files:

- Mapped CID list file **(.txt)**
- **count** output gene expression matrix file **(.raw.gef)**
- imageTools ipr2img output tissue segmentation mask TIFF file **(.tif, optional)**

Expected running time for ~1G reads: ~6 min, Memory: ~6 G

2.8.1 Arguments and Options

Table 2-9 `tissueCut` Arguments and Options

Parameter	Function
`--dnbfile`	(Optional) Mapped CID list file with reads counts for each CID. Input merged mapped CID list file if more than one list file was generated in `mapping`.
`-i`	(Required) `count` output gene expression matrix file.
`-o`	(Required) Path to the directory where to store the `tissueCut` outputs.
`-s`	(Optional) Path to the `imageTools ipr2img` output tissue segmentation mask file. Only valid when `register` has been performed.
`--sn`	(Required) Stereo-seq Chip T serial number (SN).
`--omics,-O`	(Required; default to Transcriptomics) String that specifies the omics.
`-d`	(Required) Set to generate required plots for `report`.
`-t`	(Optional) Number of threads used in numpy (recommended to fill 8 here).
`-b`	(Optional; default to 1,20,50,100,200) Set to specify bin sizes, separated by comma.
`-l`	(Optional) Set to specify the output label name.

2.8.2 Usage Example

Run `tissueCut` if `register` aligned microscopic staining image is provided:

- Generate tissue GEF based on the corresponding tissue mask image.

```
$ nucleusLayer=$(find /path/to/output/04.register -maxdepth 1 -name *fov_stitched_
transformed.tif -exec sh -c 'for f do basename -- "$f" _fov_stitched_transformed.
tif;done' sh {} + | grep -v IF)
$ tissueMaskFile=$(find /path/to/output/04.register -maxdepth 1 -name ${nucle-
usLayer}*_tissue_cut.tif

$ mkdir -p /path/to/output/05.tissuecut
$ singularity exec SAW_v7.0.sif tissueCut \
    --dnbfile /path/to/output/02.merge/{SN}.merge.barcodeReadsCount.txt \
    -i /path/to/output/03.count/{SN}.raw.gef \
    -o /path/to/output/05.tissuecut \
    -s ${tissueMaskFile} \
    --sn {SN} -O Transcriptomics -d
```

- Output GEF with a customized label.

```
$ label=<labelName> ## customized label name
$ labelMaskFile=$(find /path/to/output/04.register -maxdepth 1 -name *${la-
bel}*_tissue_cut.tif)

$ mkdir -p /path/to/output/05.tissuecut/tissuecut_${label}
$ singularity exec SAW_v7.0.sif tissueCut \
    --dnbfile /path/to/output/02.merge/{SN}.merge.barcodeReadsCount.txt \
```

```
    -i /path/to/output/03.count/{SN}.raw.gef \
    -o /path/to/output/05.tissuecut/tissuecut_${label} \
    -s ${labelMaskFile} \
    -l ${label}
    --sn {SN} -O Transcriptomics -d
```

Run **tissueCut** if image is not available:

```
$ mkdir -p /path/to/output/05.tissuecut
$ singularity exec SAW_v7.0.sif tissueCut \
    --dnbfile /path/to/output/02.merge/{SN}.merge.barcodeReadsCount.txt \
    -i /path/to/output/03.count/{SN}.raw.gef \
    -o /path/to/output/05.tissuecut \
    --sn {SN} -O Transcriptomics -d
```

2.8.3 Outputs

tissueCut output files:

Image is provided:

- Generate tissue GEF based on the corresponding tissue mask image.

```
$ tree /path/to/output/05.tissuecut
/path/to/output/05.tissuecut
├── SN.tissue.gef
├── tissuecut.stat
└── tissue_fig
    ├── scatter_100x100_MID_gene_counts.png
    ├── scatter_150x150_MID_gene_counts.png
    ├── scatter_200x200_MID_gene_counts.png
    ├── scatter_20X20_MID_gene_counts.png
    ├── scatter_50x50_MID_gene_counts.png
    ├── statistic_100x100_MID_gene_DNB.png
    ├── statistic_150x150_MID_gene_DNB.png
    ├── statistic_200x200_MID_gene_DNB.png
    ├── statistic_20X20_MID_gene_DNB.png
    ├── statistic_50x50_MID_gene_DNB.png
    ├── violin_100x100_MID_gene.png
    ├── violin_150x150_MID_gene.png
    ├── violin_200x200_MID_gene.png
    ├── violin_20X20_MID_gene.png
    └── violin_50x50_MID_gene.png
```

- Output GEF with a customized label.

```
$ tree /path/to/output/05.tissuecut/tissuecut_<label>
/path/to/output/05.tissuecut/tissuecut_<label>
...
├── tissuecut_<label>
│   ├── SN.<label>.raw.label.gef
│   ├── <label>.tissuecut.stat
│   └── tissue_fig
│       ├── scatter_100x100_MID_gene_counts.png
│       ├── scatter_150x150_MID_gene_counts.png
│       ├── scatter_200x200_MID_gene_counts.png
│       ├── scatter_20x20_MID_gene_counts.png
│       ├── scatter_50x50_MID_gene_counts.png
│       ├── statistic_100x100_MID_gene_DNB.png
│       ├── statistic_150x150_MID_gene_DNB.png
│       ├── statistic_200x200_MID_gene_DNB.png
│       ├── statistic_20x20_MID_gene_DNB.png
│       ├── statistic_50x50_MID_gene_DNB.png
│       ├── violin_100x100_MID_gene.png
│       ├── violin_150x150_MID_gene.png
│       ├── violin_200x200_MID_gene.png
│       ├── violin_20x20_MID_gene.png
│       └── violin_50x50_MID_gene.png
...
```

Image is not provided:

```
$ tree /path/to/output/05.tissuecut
/path/to/output/05.tissuecut
├── 100X100_contour_image.png  ## bin100 expression png
├── bin1_img.tif  ## bin1 expresion distribution TIFF file
├── bin1_img_tissue_cut.tif  ## tissue mask acquried from bin1 expression distribu-
tion plot
├── SN.tissue.gef
├── tissuecut.stat
└── tissue_fig
    ├── scatter_100x100_MID_gene_counts.png
    ├── scatter_150x150_MID_gene_counts.png
    ├── scatter_200x200_MID_gene_counts.png
    ├── scatter_20X20_MID_gene_counts.png
    ├── scatter_50x50_MID_gene_counts.png
    ├── statistic_100x100_MID_gene_DNB.png
    ├── statistic_150x150_MID_gene_DNB.png
    ├── statistic_200x200_MID_gene_DNB.png
    ├── statistic_20X20_MID_gene_DNB.png
    ├── statistic_50x50_MID_gene_DNB.png
    ├── violin_100x100_MID_gene.png
    ├── violin_150x150_MID_gene.png
    ├── violin_200x200_MID_gene.png
    ├── violin_20X20_MID_gene.png
    └── violin_50x50_MID_gene.png
```

2.9 spatialCluster

SAW **spatialCluster** pipeline performs clustering analysis for spots using Stereopy. The clustering procedure includes 4 main steps: 1) preprocess gene expression data from the tissue-coverage region (normalize, logarithmize, identify highly-variable genes, and scale each gene), 2) reduce the dimensionality of the data by running PCA, 3) compute the neighborhood graph and embed neighborhood graph using UMAP, 4) clustering by Leiden algorithm.

Running **spatialCluster** requires the following files:

- **tissueCut** output GEF file for the tissue-covered region **(.tissue.gef)**

Expected running time for ~1G reads: ~1 min, Memory: ~5 G

2.9.1 Arguments and Options

Table 2-10 **spatialCluster** Arguments and Options

Parameter	Function
-i	(Required) **tissueCut** output GEF file for the tissue coverage area.
-o	(Required) Output path for the clustering result in H5AD format.
-s	(Required; default to 50) Bin size.
-r	(Required; default to 1.0) Leiden resolution which controls the coarseness of the clustering. Higher values lead to more clusters. Leiden resolution is a floating value in the range of 0.1 to 1.

2.9.2 Usage Example

```
$ mkdir -p /path/to/output/06.spatialcluster
$ singularity exec SAW_v7.0.sif spatialCluster \
    -i /path/to/output/05.tissuecut/{SN}.tissue.gef \
    -r 1.0 \
    -s 200 \
    -o /path/to/output/06.spatialcluster/{SN}.spatial.cluster.h5ad
    ## SN is suffixed with bin_size and resolution information
```

2.9.3 Outputs

spatialCluster output files are:

```
$ tree /path/to/output/06.spatialcluster
/path/to/output/06.spatialcluster
└── SN.bin200_1.0.spatial.cluster.h5ad
```

2.10 cellCut

SAW **cellCut** pipeline runs to extract expression matrix of cell nucleus based on the aligned image generated from **register** and **imageTools**. **cellCut** outputs expression data in cell bin GEF format. Run **cellCut** if cell bin results are desired.

Running **cellCut** requires the following files:

- **count** output gene expression matrix file **(.raw.gef)**
- **register** and **imageTools** output cell segmentation mask TIFF file **(.tif)**

Expected running time for ~1G reads: ~2 min, Memory: ~10 G

2.10.1 Arguments and Options

Table 2-11 **cellCut** Arguments and Options

Commands	Parameters	Function
cgef	-i, --input-file	(Required) Input GEF file.
	-m, --mask-file	(Required) Input cell segmentation mask file.
	-o, --output-file	(Required) Output cell bin GEF file.
	-b, --block	(Optional; default to 256,256) Block size.
	-r, --rand-celltype	(Optional; default to 0) Number of random cell type.
	-t, --threads	(Optional; default to 1) Number of threads.
	-v, --verbose	(Optional) Verbose output.
	-n, --cnum	(Optional; default to 5000) Top level cell number.
	-R, --ratio	(Optional; default to 20) Other level cell number ratio.
	-a, --allocat	(Optional; default to 2) Allocation strategy.
	-g, --raw-gem	(Optional) Raw GEM file.
	-c, --canvas	(Optional; default to 0,0,90000,90000) Set canvas size, minX,minY,-maxX maxY.
	-l, --limit	(Optional; default to 16,16) Set block limit.
	-S, --split	(Optional; default to 0) Split cellID to layers and blocks.
	-w, --errorCode-file	(Optional; default to false) Set to turn on error code mode.

Please check 2.14.1 Other applications of cellCut to learn more about cellCut.

2.10.2 Usage Example

Run **cellCut** only if **register** aligned microscopic staining image is provided:

```
$ mkdir -p /path/to/output/051.cellcut
$ nucleusLayer=$(find /path/to/output/04.register -maxdepth 1 -name *fov_stitched_
transformed.tif -exec sh -c 'for f do basename -- "$f" _fov_stitched_transformed.
tif;done' sh {} + | grep -v IF)
$ nucleusMask=$(find /path/to/output/04.register -maxdepth 1 -name ${nucleusLayer}*_
mask.tif)

$ singularity exec SAW_v7.0.sif cellCut cgef \
      -i /path/to/output/03.count/{SN}.raw.gef \
      -m ${nucleusMask} \
      -o /path/to/output/051.cellcut/{SN}.cellbin.gef
```

2.10.3 Outputs

cellCut output files:

```
$ tree /path/to/output/051.cellcut
/path/to/output/051.cellcut
└── SN.cellbin.gef
```

2.11 cellCorrect

SAW **cellCorrect** pipeline runs to make adjustments based on the aligned cell segmentation image generated from **register** and **imageTools** and and extract expression matrix of the adjusted one. **cellCorrect** outputs expression data in cell bin GEF and GEM formats. Run **cellCorrect** if results of cell correction are desired.

Running **cellCorrect** requires the following files:

- **count** output gene expression matrix file **(.raw.gef)**
- **register** and imageTools output cell segmentation mask TIFF file (.tif)

Expected running time for ~1G reads: ~2 min, Memory: ~10 G

2.11.1 Arguments and Options

Table 2-12 **cellCluster** Arguments and Options

Parameter	Function
-i	(Required) Input GEF file.
-m	(Required) Input cell segmentation mask file.
-d	(Required; default to 10) Outspread distance based on the cellular contour of cell segmentation image, in pixels.
-o	(Required) Path to output cell bin GEF, GEM and adjusted mask files.

2.11.2 Usage Example

Run **cellCorrect** as below:

```
$ nucleusLayer=$(find /path/to/output/04.register -maxdepth 1 -name *fov_stitched_
transformed.tif -exec sh -c 'for f do basename -- "$f" _fov_stitched_transformed.
tif;done' sh {} + | grep -v IF)
$ nucleusMask=$(find /path/to/output/04.register -maxdepth 1 -name ${nucleusLayer}*_
mask.tif)

$ singularity exec SAW_v7.0.sif cellCorrect \
      -i /path/to/output/03.count/{SN}.raw.gef \
      -m ${nucleusMask} \
      -d 10 \
      -o /path/to/output/051.cellcut
```

2.11.3 Outputs

cellCorrect output files:

```
$ tree /path/to/output/051.cellcut
/path/to/output/051.cellcut
├── SN.adjusted.cellbin.gef
├──<stainType>_SN_mask_edm_dis_10.tif
└── SN.adjusted.gem
```

2.12 cellCluster

SAW **cellCluster** pipeline runs by clustering cells using the Leiden algorithm similarly to the procedures of **spatialCluster**. Run **cellCluster** if cell bin results are desired.

Running **cellCluster** requires the following files:

- **cellCut** output cell bin gene expression matrix file **(.cellbin.gef)**

Expected running time for ~1G reads: ~5 min, Memory: ~5 G

2.12.1 Arguments and Options

Table 2-12 cellCluster Arguments and Options

Parameter	Function
-i	(Required) **cellCut** output cell bin gene expression matrix file.
-o	(Required) Output path to the clustering result in H5AD format.

2.12.2 Usage Example

```
$ mkdir -p /path/to/output/061.cellcluster
$ singularity exec SAW_v7.0.sif cellCluster \
        -i /path/to/output/051.cellcut/{SN}.adjusted.cellbin.gef \
        -o /path/to/output/061.cellcluster/{SN}.adjusted.cell.cluster.h5ad

$ singularity exec SAW_v7.0.sif cellCluster \
        -i /path/to/output/051.cellcut/{SN}.cellbin.gef \
        -o /path/to/output/061.cellcluster/{SN}.cell.cluster.h5ad
```

2.12.3 Outputs

cellCluster output files:

```
$ tree /path/to/output/061.cellcluster
/path/to/output/061.cellcluster
├── SN.adjusted.cell.cluster.h5ad
└── SN.cell.cluster.h5ad
```

2.13 saturation

SAW **saturation** pipeline is performed to compute the sequencing saturation for the tissue coverage area.

Running **saturation** requires the following files:

- **mapping** output statistical report of CID mapping **(.stat)**
- **count** output saturation sampling file **(.txt)**
- **count** output statistical report of annotation **(.stat)**
- **tissueCut** output GEF file for the tissue coverage area **(.tissue.gef)**

Expected running time for ~1G reads: ~5 min, Memory: ~5 G

2.13.1 Arguments and Options

Table 2-13 **saturation** Arguments and Options

Parameter	Function
-i	(Required) **count** output saturation sampling file.
--tissue	(Required) **tissueCut** output GEF file for the tissue coverage area.
-o	(Required) Path to the directory where to store the **saturation** outputs. There has to be an existing path.
--bcstat	(Required) **mapping** output statistical report of CID mapping. Separate multiple files by comma.
--summary	(Required) **count** output statistical report of annotation.

2.13.2 Usage Example

```
$ mkdir -p /path/to/output/07.saturation
$ singularity exec SAW_v7.0.sif saturation \
      -i /path/to/output/03.count/{SN}_raw_barcode_gene_exp.txt \
      --tissue /path/to/output/05.tissuecut/{SN}.tissue.gef \
      -o /path/to/output/07.saturation \
      --bcstat /path/to/output/01.mapping/{lane}.CIDMap.stat \
      --summary  /path/to/output/03.count/{SN}.Aligned.sortedByCoord.out.merge.q10.
dedup.target.bam.summary.stat
```

* Please be noted that these two lines belong to the same line of command.

For more than one pair of FASTQ files (Here showing an example of 2 pairs of FASTQ),

```
$ mkdir -p /path/to/multi_lane_output/07.saturation
$ singularity exec SAW_v7.0.sif saturation \
    -i /path/to/multi_lane_output/03.count/{SN}_raw_barcode_gene_exp.txt \
    --tissue /path/to/multi_lane_output/05.tissuecut/{SN}.tissue.gef \
    -o /path/to/multi_lane_output/07.saturation \
    --bcstat /path/to/multi_lane_output/01.mapping/{lane1}.CIDMap.stat,/path/to/
multi_lane_output/01.mapping/{lane2}.CIDMap.stat \
    --summary /path/to/multi_lane_output/03.count/{SN}.Aligned.sortedByCoord.out.
merge.q10.dedup.target.bam.summary.stat
```

* Please be noted that we use the backward slash "\" to indicate the end of a line in a command that spans multiple lines.

2.13.3 Outputs

saturation output files:

```
$ tree /path/to/output/07.saturation
├── plot_1x1_saturation.png
├── plot_200x200_saturation.png
└── sequence_saturation.tsv
```

2.14 report

SAW `report` pipeline is performed to integrate analysis results from each step and generate the report in JSON format as well as a web report in HTML format. HTML analytical report integrate genes' spatial expression distribution, key statistical metrices, sequencing saturation plots, clustering analysis results, and image processing information. When it comes to the scenario of mIF, pseudo color (up to 7) is used on the bottom layer of clustering results.

Running `report` requires the following files and information:

- `mapping` output statistical report of CID mapping and **STAR** alignment **(.stat, .out)**
- `count` output statistical report of annotation **(.stat)**
- `register` processed image process record file and image pyramid RPI file **(.ipr, .rpi)**
- `tissueCut` and `cellCut` (if available) output GEF file, statistical report of tissue-covered region, plots and image pyramid RPI file **(.gef, .stat, .png)**
- `spatialCluster` and `cellCluster` (if available) output clustering H5AD file **(.h5ad)**
- `saturation` output bin200 sequence saturation plot **(.png)**
- species, tissue, and reference information

Expected running time for ~1G reads: ~1 min, Memory: 1G

2.14.1 Arguments and Options

Table 2-14 `report` Arguments and Options

Parameter	Function
`-m`	(Required) Statistical report of CID mapping. Separate multiple files by comma.
`-a`	(Required) Statistical report of STAR alignment. Separate multiple files by comma.
`-g`	(Required) Statistical report of annotation.
`-l`	(Required) Statistical report of tissue-covered region.
`-n`	(Required) GEF file that has wholeExp/bin200. Please check **2.14.1 Other applications of cellCut** to get more information of GEF completion.
`-b`	(Required) `tissueCut` output bin 200 scatter plot.
`-v`	(Required) `tissueCut` output bin 200 violin plot.
`-c`	(Required) `tissueCut` output bin 200 statistics plot.
`--bin20Saturation`	(Required) `tissueCut` output bin 20 scatter plot.
`--bin20violin`	(Required) `tissueCut` output bin 20 violin plot.
`--bin20MIDGeneDNB`	(Required) `tissueCut` output bin 20 statistics plot.
`--bin50Saturation`	(Required) `tissueCut` output bin 50 scatter plot.
`--bin50violin`	(Required) `tissueCut` output bin 50 violin plot.
`--bin50MIDGeneDNB`	(Required) `tissueCut` output bin 50 statistics plot.
`--bin100Saturation`	(Required) `tissueCut` output bin 100 scatter plot.
`--bin100violin`	(Required) `tissueCut` output bin 100 violin plot.
`--bin100MIDGeneDNB`	(Required) `tissueCut` output bin 100 statistics plot.
`--bin150Saturation`	(Required) `tissueCut` output bin 150 scatter plot.

Parameter	Function
--bin150violin	(Required) `tissueCut` output bin 150 violin plot.
--bin150MIDGeneDNB	(Required) `tissueCut` output bin 150 statistics plot.
-d	(Required) `spatialCluster` output H5AD file.
-t	(Required) `saturation` output bin 200 sequence saturation plot.
-r standard_version	(Required) Set to specifying report version.
-s	(Required) The Stereo-seq Chip T serial number.
--pipelineVersion	(Required) Set to specifying analysis pipeline version.
-o	(Required) The directory to store outputs.
--species	(Required) A string of species name.
--tissue	(Required) A string of tissue type.
-reference	(Required) A string of reference used for `mapping`
--rpi_resolution	(Optional; default to 100) The resolution of RPI. Valid options: **2, 10, 50, 100, 150**.
-i	(Optional) The image pyramid RPI file.
--iprFile	(Optional) A processed IPR (image process record) file.
--cellBinGef	(Optional) The cell bin GEF file.
--cellCluster	(Optional) `cellCluster` output H5AD file.
--tissue_fig	(Optioanal) The directory of statistical images from `tissueCut`.
--qc_metrics	(Optional) The customized QC-metrics file (.xlsx) for SAW. QC-metrics focus on sequencing quality, and performance of in situ restoration of sequencing reads. For example, "Total Q30" and "Fraction MID in Spots Under Tissue".

2.14.2 Usage Example

Run **report** if **register** aligned microscopic staining image is provided and has cell bin output:

Please Note! Replace {SN}, {lane},{species_name}, {tissue_type}, {reference_index} with the real information.

```
$ imageIPR=$(find /path/to/output/04.register -maxdepth 1 -name {SN}*.ipr | head
-1)  ## has to be a processed IPR
$ singularity exec SAW_v7.0.sif cellCut bgef \
    -i /path/to/output/03.count/{SN}.raw.gef \
    -o /path/to/output/05.tissuecut/{SN}.gef \
    -b 1,10,20,50,100,200,500 \
    -O Transcriptomics

$ mkdir -p /path/to/output/08.report
$ singularity exec SAW_v7.0.sif report \
      -m /path/to/output/01.mapping/{lane}.CIDMap.stat \
       -a /path/to/output/01.mapping/{lane}.Log.final.out \
      -g /path/to/output/03.count/{SN}.Aligned.sortedByCoord.out.merge.q10.dedup.
target.bam.summary.stat \
      -l /path/to/output/05.tissuecut/tissuecut.stat \
      -n /path/to/output/05.tissuecut/{SN}.gef \
```

* Please be noted that we use the backward slash "\" to indicate the end of a line in a command that spans multiple lines.

```
    gene_counts.png      \,--bin20violin,--bin20MIDGeneDNB
     -d /path/to/output/06.spatialcluster/{SN}.bin200_1.0.spatial.cluster.h5ad \
     -t /path/to/output/07.saturation/plot_200x200_saturation.png \
     -b /path/to/output/05.tissuecut/tissue_fig/scatter_200x200_MID_gene_counts.
png \
     -v /path/to/output/05.tissuecut/tissue_fig/violin_200x200_MID_gene.png \
     -c /path/to/output/05.tissuecut/tissue_fig/statistic_200x200_MID_gene_DNB.
png \
     --bin20Saturation /path/to/output/05.tissuecut/tissue_fig/scatter_20x20_MID_
gene_counts.png \
     --bin20violin /path/to/output/05.tissuecut/tissue_fig/violin_20x20_MID_gene.
png \
     --bin20MIDGeneDNB /path/to/output/05.tissuecut/tissue_fig/statistic_20x20_
MID_gene_DNB.png \
     --bin50Saturation /path/to/output/05.tissuecut/tissue_fig/scatter_50x50_MID_
gene_counts.png \
     --bin50violin /path/to/output/05.tissuecut/tissue_fig/violin_50x50_MID_gene.
png \
     --bin50MIDGeneDNB /path/to/output/05.tissuecut/tissue_fig/statistic_50x50_
MID_gene_DNB.png \
     --bin100Saturation /path/to/output/05.tissuecut/tissue_fig/scatter_100x100_
MID_gene_counts.png \
     --bin100violin /path/to/output/05.tissuecut/tissue_fig/violin_100x100_MID_
gene.png \
     --bin100MIDGeneDNB /path/to/output/05.tissuecut/tissue_fig/statis-
tic_100x100_MID_gene_DNB.png \
     --bin150Saturation /path/to/output/05.tissuecut/tissue_fig/scatter_150x150_
MID_gene_counts.png \
     --bin150violin /path/to/output/05.tissuecut/tissue_fig/violin_150x150_MID_
gene.png \
     --bin150MIDGeneDNB /path/to/output/05.tissuecut/tissue_fig/statis-
tic_150x150_MID_gene_DNB.png \
     --cellBinGef /path/to/output/051.cellcut/{SN}.adjusted.cellbin.gef \
     --cellCluster /path/to/output/061.cellcluster/{SN}.adjusted.cell.cluster.h5ad \
     -i /path/to/output/04.register/{SN}.rpi \
     -r standard_version \
     -s {SN} \
     --pipelineVersion SAW_v7.0.0 \
     --iprFile ${imageIPR} \
     --species {species_name} \
     --tissue {tissue_type} \
     --reference {reference_index} \
     -o /path/to/output/08.report
```

* Please be noted that we use the backward slash "\" to indicate the end of a line in a command that spans multiple lines.

For more than one pair of FASTQ files (Here showing an example of 2 pairs of FASTQ),

```
$ mkdir -p /path/to/multi_lane_output/08.report
$ singularity exec SAW_v7.0.sif cellCut bgef \
    -i /path/to/output/03.count/{SN}.raw.gef \
    -o /path/to/output/05.tissuecut/{SN}.gef \
    -b 1,10,20,50,100,200,500 \
    -O Transcriptomics
$ singularity exec SAW_v7.0.sif report \
      -m /path/to/multi_lane_output/01.mapping/{lane1}.CIDMap.stat,/path/to/multi_
lane_output/01.mapping/{lane2}.CIDMap.stat \
      -a /path/to/multi_lane_output/01.mapping/{lane1}.Log.final.out,/path/to/
multi_lane_output/01.mapping/{lane2}.Log.final.out \
      -g /path/to/multi_lane_output/03.count/{SN}.Aligned.sortedByCoord.out.merge.
q10.dedup.target.bam.summary.stat \
      -l /path/to/multi_lane_output/05.tissuecut/tissuecut.stat \
      -n /path/to/multi_lane_output/05.tissuecut/{SN}.gef \
      -d /path/to/multi_lane_output/06.spatialcluster/{SN}.bin200_1.0.spatial.
cluster.h5ad \
      -t /path/to/multi_lane_output/07.saturation/plot_200x200_saturation.png \
      -b /path/to/multi_lane_output/05.tissuecut/tissue_fig/scatter_200x200_MID_
gene_counts.png \
      -v /path/to/multi_lane_output/05.tissuecut/tissue_fig/violin_200x200_MID_
gene.png \
      -c /path/to/multi_lane_output/05.tissuecut/tissue_fig/statistic_200x200_MID_
gene_DNB.png \
       --bin20Saturation /path/to/mult_lane_output/05.tissuecut/tissue_fig/scat-
ter_20x20_MID_gene_counts.png \
       --bin20violin /path/to/multi_lane_output/05.tissuecut/tissue_fig/vio-
lin_20x20_MID_gene.png \
       --bin20MIDGeneDNB /path/to/multi_lane_output/05.tissuecut/tissue_
fig/statistic_20x20_MID_gene_DNB.png \
      --bin50Saturation /path/to/mult_lane_output/05.tissuecut/tissue_fig/scat-
ter_50x50_MID_gene_counts.png \
      --bin50violin /path/to/multi_lane_output/05.tissuecut/tissue_fig/vio-
lin_50x50_MID_gene.png \
      --bin50MIDGeneDNB /path/to/multi_lane_output/05.tissuecut/tissue_
fig/statistic_50x50_MID_gene_DNB.png \
      --bin100Saturation /path/to/multi_lane_output/05.tissuecut/tissue_
fig/scatter_100x100_MID_gene_counts.png \
      --bin100violin /path/to/multi_lane_output/05.tissuecut/tissue_
fig/violin_100x100_MID_gene.png \
      --bin100MIDGeneDNB /path/to/multi_lane_output/05.tissuecut/tissue_
fig/statistic_100x100_MID_gene_DNB.png \
      --bin150Saturation /path/to/multi_lane_output/05.tissuecut/tissue_
fig/scatter_150x150_MID_gene_counts.png \
```

```
      --bin150violin /path/to/multi_lane_output/05.tissuecut/tissue_
fig/violin_150x150_MID_gene.png \
      --bin150MIDGeneDNB /path/to/multi_lane_output/05.tissuecut/tissue_
fig/statistic_150x150_MID_gene_DNB.png \
      --cellBinGef /path/to/output/051.cellcut/{SN}.adjusted.cellbin.gef \
      --cellCluster /path/to/output/061.cellcluster/{SN}.adjusted.cell.cluster.h5ad
      -i /path/to/multi_lane_output/04.register/{SN}.rpi \
      -r standard_version \
      -s {SN} \
      --pipelineVersion SAW_v7.0.0 \
      --iprFile ${imageIPR} \
      --species {species_name} \
      --tissue {tissue_type} \
      --reference {reference_index} \
      -o /path/to/multi_lane_output/08.report
```

* Please be noted that we use the backward slash "\" to indicate the end of a line in a command that spans multiple lines.

Run **report** of outputs in which the microscopic image has been registered with gene expression matrix but did not perform cell segmentation. Here is an example of just one pair of FASTQ,

```
$ mkdir -p /path/to/output/08.report
$ singularity exec SAW_v7.0.0.sif cellCut bgef \
    -i /path/to/output/03.count/{SN}.raw.gef \
    -o /path/to/output/05.tissuecut/{SN}.gef \
    -b 1,10,20,50,100,200,500 \
    -O Transcriptomics

$ singularity exec SAW_v7.0.0.sif report \
    -m /path/to/output/01.mapping/{lane}.CIDMap.stat \
    -a /path/to/output/01.mapping/{lane}.Log.final.out \
    -g /path/to/output/03.count/{SN}.Aligned.sortedByCoord.out.merge.q10.dedup.
target.bam.summary.stat \
    -l /path/to/output/05.tissuecut/tissuecut.stat \
    -n /path/to/output/05.tissuecut/{SN}.gef \
    -d /path/to/output/06.spatialcluster/{SN}.bin200_1.0.spatial.cluster.h5ad \
    -t /path/to/output/07.saturation/plot_200x200_saturation.png \
    -b /path/to/output/05.tissuecut/tissue_fig/scatter_200x200_MID_gene_counts.png
\
    -v /path/to/output/05.tissuecut/tissue_fig/violin_200x200_MID_gene.png \
    -c /path/to/output/05.tissuecut/tissue_fig/statistic_200x200_MID_gene_DNB.png
\
    --bin20Saturation /path/to/output/05.tissuecut/tissue_fig/scatter_20x20_MID_
gene_counts.png \
    --bin20violin /path/to/output/05.tissuecut/tissue_fig/violin_20x20_MID_gene.
png \
    --bin20MIDGeneDNB /path/to/output/05.tissuecut/tissue_fig/statistic_20x20_MID_
gene_DNB.png \
```

* Please be noted that we use the backward slash "\" to indicate the end of a line in a command that spans multiple lines.

```
    --bin50Saturation /path/to/output/05.tissuecut/tissue_fig/scatter_50x50_MID_
gene_counts.png \
    --bin50violin /path/to/output/05.tissuecut/tissue_fig/violin_50x50_MID_gene.
png \
    --bin50MIDGeneDNB /path/to/output/05.tissuecut/tissue_fig/statistic_50x50_MID_
gene_DNB.png \
    --bin100Saturation /path/to/output/05.tissuecut/tissue_fig/scatter_100x100_
MID_gene_counts.png \
    --bin100violin /path/to/output/05.tissuecut/tissue_fig/violin_100x100_MID_
gene.png \
    --bin100MIDGeneDNB /path/to/output/05.tissuecut/tissue_fig/statistic_100x100_
MID_gene_DNB.png \
    --bin150Saturation /path/to/output/05.tissuecut/tissue_fig/scatter_150x150_
MID_gene_counts.png \
    --bin150violin /path/to/output/05.tissuecut/tissue_fig/violin_150x150_MID_gene.
png \
    --bin150MIDGeneDNB /path/to/output/05.tissuecut/tissue_fig/statistic_150x150_
MID_gene_DNB.png \
    -i /path/to/output/04.register/{SN}.rpi \
    -r standard_version \
    -s {SN} \
    --pipelineVersion SAW_v7.0.0 \
    --iprFile ${imageIPR} \
    --species {species_name} \
    --tissue {tissue_type} \
    --reference {reference_index} \
    -o /path/to/output/08.report
```

Run **report** for **register** aligned microscopic staining image is absent (Here shows an example of just one pair of FASTQ, similar to multiple pairs),

```
$ mkdir -p /path/to/output/08.report
$ singularity exec SAW_v7.0.sif cellCut bgef \
    -i /path/to/output/03.count/{SN}.raw.gef \
    -o /path/to/output/05.tissuecut/{SN}.gef \
    -b 1,10,20,50,100,200,500 \
    -O Transcriptomics
$ singularity exec SAW_v7.0.sif report \
      -m /path/to/output/01.mapping/{lane}.CIDMap.stat \
      -a /path/to/output/01.mapping/{lane}.Log.final.out \
      -g /path/to/output/03.count/{SN}.Aligned.sortedByCoord.out.merge.q10.dedup.
target.bam.summary.stat \
```

* Please be noted that we use the backward slash "\" to indicate the end of a line in a command that spans multiple lines.

```
      -l /path/to/output/05.tissuecut/tissuecut.stat \
      -n /path/to/output/05.tissuecut/{SN}.gef \
      -d /path/to/output/06.spatialcluster/{SN}.bin200_1.0.spatial.cluster.h5ad \
      -t /path/to/output/07.saturation/plot_200x200_saturation.png \
      -b /path/to/output/05.tissuecut/tissue_fig/scatter_200x200_MID_gene_counts.
png \
      -v /path/to/output/05.tissuecut/tissue_fig/violin_200x200_MID_gene.png \
      -c /path/to/output/05.tissuecut/tissue_fig/statistic_200x200_MID_gene_DNB.
png \
      --bin20Saturation /path/to/output/05.tissuecut/tissue_fig/scatter_20x20_MID_
gene_counts.png \
      --bin20violin /path/to/output/05.tissuecut/tissue_fig/violin_20x20_MID_gene.
png \
      --bin20MIDGeneDNB /path/to/output/05.tissuecut/tissue_fig/statistic_20x20_
MID_gene_DNB.png \
      --bin50Saturation /path/to/output/05.tissuecut/tissue_fig/scatter_50x50_MID_
gene_counts.png \
      --bin50violin /path/to/output/05.tissuecut/tissue_fig/violin_50x50_MID_gene.
png \
      --bin50MIDGeneDNB /path/to/output/05.tissuecut/tissue_fig/statistic_50x50_
MID_gene_DNB.png \
      --bin100Saturation /path/to/output/05.tissuecut/tissue_fig/scatter_100x100_
MID_gene_counts.png \
      --bin100violin /path/to/output/05.tissuecut/tissue_fig/violin_100x100_MID_
gene.png \
      --bin100MIDGeneDNB /path/to/output/05.tissuecut/tissue_fig/statis-
tic_100x100_MID_gene_DNB.png \
      --bin150Saturation /path/to/output/05.tissuecut/tissue_fig/scatter_150x150_
MID_gene_counts.png \
      --bin150violin /path/to/output/05.tissuecut/tissue_fig/violin_150x150_MID_
gene.png \
      --bin150MIDGeneDNB /path/to/output/05.tissuecut/tissue_fig/statis-
tic_150x150_MID_gene_DNB.png \
      -r standard_version \
      -s {SN} \
      --pipelineVersion SAW_v7.0.0 \
      --species {species_name} \
      --tissue {tissue_type} \
      --reference {reference_index} \
      -o /path/to/output/08.report
```

* Please be noted that we use the backward slash "\" to indicate the end of a line in a command that spans multiple lines.

2.14.3 Outputs

report output files that have cell bin data input are organized as below:

```
$ tree /path/to/output/08.report
/path/to/output/08.report
├── scatter_1x1_MID_gene_counts.png
├── SN.report.html
├── SN.statistics.json
├── statistic_1x1_cell_area.png
├── statistic_1x1_DNB.png
├── statistic_1x1_gene.png
├── statistic_1x1_MID.png
├── violin_1x1_gene.png
└── violin_1x1_MID.png
```

report output files that the cell bin data inputs are absent:

```
$ tree /path/to/output/08.report
/path/to/output/08.report
├── SN.report.html
└── SN.statistics.json
```

2.15 Other Applications

2.15.1 Other applications of cellCut

SAW **cellCut** is also an application for manipulating GEF file. SAW contains this tool to convert GEF format gene expression matrix to plain table or complete a GEF. Users may also manipulate the GEF files using the separate individual C++ compiled program geftools[12] or its python encapsulated package gefpy[13].

2.15.1.1 Arguments and Options

Table 2-15 Other applications of `cellCut` Arguments and Options

Commands	Parameters	Function
view	-i, --input-file	(Required) Input bin GEF file or cell bin GEF file.
	-o, --output-file	(Optional; default to stdout) Output GEM file.
	-d, --exp_data	(Optional; default to “”) Input bin1 GEF to get cell bin GEM.
	-m, --mask-file	(Optional; default to "") Input cell segmentation mask file, when cell bin GEM is expected output.
	-b, --bin-size	(Optional; default to1) Set bin size for bin GEF file. Only valid for bin GEF.
	-s, --serial-number	(Required) Stereo-seq Chip T serial number.
	-e, --exon	(Optional; default to 1) Whether or not output exon group.
	-w, --errorCode-file	(Optional; default to false) Determine output error code to file.
bgef	-i, --input-file	(Required) Input gene expression matrix file (.gem/.gem.gz) or bin1 GEF file.
	-o, --output-file	(Required) Output bin GEF file.
	-b, --bin-size	(Required; default to 1,10,20,50,100,200,500) Comma-separate bin size list.
	-r, --region	(Optional; default to “”) A rectangular region represented by the comma-separated list of vertex coordinates. For example, minx,maxX-,minY,maxY.
	-t, --threads	(Optional; default to 8) Number of threads.
	-s, --stat	(Optional; default to true) Whether create stat group. Stat group includes a gene dataset which contains total MIDcount and E10 score for each gene.
	-O, --omics	(Required; default to Transcriptomics) Specify the omics.
	-v, --verbose	(Optional) Verbose output.
	-w, --errorCode-file	(Optional; default to false) Determine output error code to file.
cgef	-i, --input-file	(Required) Input GEF file.
	-m, --mask-file	(Required) Input cell segmentation mask file.
	-o, --output-file	(Required) Output cell bin GEF file.
	-b, --block	(Optional; default to 256,256) Block size.
	-r, --rand-celltype	(Optional; default to 0) Number of random cell type.
	-t, --threads	(Optional; default to 1) Number of threads.
	-v, --verbose	(Optional) Verbose output.
	-n, --cnum	(Optional; default to 5000) Top level cell number.
	-R, --ratio	(Optional; default to 20) Other level cell number ratio.
	-a, --allocat	(Optional; default to 2) Allocation strategy.
	-g, --raw-gem	(Optional) Raw GEM file.
	-c, --canvas	(Optional; default to 0,0,90000,90000) Set canvas size, minX, minY, maxX, maxY.
	-l, --limit	(Optional; default to 16,16) Set block limit.
	-S, --split	(Optional; default to 0) Split cellID to layers and blocks.
	-w, --errorCode-file	(Optional; default to false) Set to turn on error code mode.

2.14.1.2 Usage Examples

Function 1: GEF to plain table GEM format

```
$ singularity exec SAW_v7.0.sif cellCut view \  ## convert GEF that only contains
bin1 geneExp
      -s {SN} \
      -i /path/to/output/03.count/{SN}.raw.gef \
      -o {SN}.raw.gem
$ singularity exec SAW_v7.0.sif cellCut view \  ## convert a whole GEF
      -s {SN} \
      -i /path/to/output/05.tissuecut/{SN}.gef \
      -o {SN}.gem
$ singularity exec SAW_v7.0.sif cellCut view \  ## convert tissue GEF that only
contains bin1 geneExp
      -s {SN} \
      -i /path/to/output/05.tissuecut/{SN}.tissue.gef \
      -o {SN}.tissue.gem
$ singularity exec SAW_v7.0.sif cellCut view \  ## convert cellbin GEF to cellbin
GEM
      -s {SN} \
      -i /path/to/output/051.cellcut/{SN}.cellbin.gef \
      -o {SN}.cellbin.gem \
      -d /path/to/output/03.count/{SN}.raw.gef
```

Function 2: completion of a GEF

```
$ singularity exec SAW_v7.0.sif cellCut bgef \  ## complete GEF that only contains
bin1 geneExp group to a whole GEF, you may specify the bin size using "-b". Sepa-
rate multiple bin size with comma
      -i /path/to/output/03.count/{SN}.tissue.gef \
      -o {SN}.tissue.complete.gef \
      -b 1,20,50,100 \
      -O Transcriptomics
```

Function 3: converting GEM to GEF

```
$ singularity exec SAW_v7.0.sif cellCut bgef \  ## convert GEM to GEF in specific
bin size. Separate multiple bin sizes with comma
      -i {SN}.gem \
      -o {SN}.gef \
      -b 1,20,50 \
      -O Transcriptomics
```

Example of GEF to GEM conversion using `gefpy`, users may specify the bin size.

```
$ python
>>> from gefpy.bgef_reader_cy import BgefR
>>> bgef=BgefR(filepath='/path/to/output/05.tissuecut/{SN}.tissue.gef',bin_
size=200,n_thread=4)
>>> bgef.to_gem('{SN}.tissue.bin200.gem')
```

2.14.2 checkGTF

SAW **checkGTF** is an application for checking whether the GTF/GFF file is in the correct format as the input for **count**, especially the limitation to the length of gene names.

Running **checkGTF** requires the following files:

- Reference genome annotation GFF/GTF[9,10] file **(.gff / .gtf)**

2.14.2.1 Arguments and Options

Table 2-17 checkGTF Arguments and Options

Parameter	Function
-i	(Required) Gene annotation GFF/GTF file.
-o	(Required) Output a re-format GFF/GTF file.

2.14.2.2 Usage Examples

```
$ singularity exec SAW_v7.0.sif checkGTF \
      -i /path/to/reference/genes.gtf \
      -o /path/to/reference/genes_new.gtf \
```

2.14.3 Other applications of imageTools

Besides **ipr2img**, **imageTools** has three more functions: **img2rpi**(writes images in RPI format), **merge** (merges stain image with tissue segmentation and cell segmentation images to check the segmentation result), and **overlay** (stacks inferred track line template with stitched panoramic image to check stitching result or stacks track line template from matrix with registered panoramic image to check registration result).

2.14.3.1 Arguments and Options

Table 2-18 Other applications of `imageTools` Arguments and Options

Commands	Parameters	Function
`img2rpi`	`-i`	(Required) Input TIFF file. Separate multiple files by comma.
	`-g`	(Required) Set the '<stainType>/<imageType>' of the input TIFF stores in the RPI. Separate multiple files by comma, the order of groups needs to be exactly same with the input TIFF files.
	`-b`	(Required) Bin sizes. Separate multiple bin sizes with space, for example 1 10 50 100
	`-o`	(Required) Output path for the RPI file. Has to use absolute path or relative path.
	`-c`	(Optional) Customized mask colors in RGB format only valid for (/<stainType>/TissueMask_<label>/Color. The default color is set to be (255,255,255) for one mask. Separate multiple mask colors by comma and and add escape characters before brackets. For example: -c \(10,20,30\),\(30,40,50\).
`merge`	`-i`	(Required) Input TIFF files. Separate multiple files by comma. The maximum number of TIFF files to be composited is three, and the channel order is R-G-B.
	`-o`	(Required) Output path for the multichannel image.
`overlay`	`-i`	(Required) Input TIFF file. Usually input a pre-registered panoramic image if overlaid with the stitching template derived from track cross, and input a registered microscopic image if overlaid with the track line template acquired from gene expression matrix.
	`-c`	(Required) Image configuration file (.ipr) or template points file (.txt).
	`-o`	(Required) Output path of the TIFF file which has the template overlaid on the selected image.
`ipr2img`	`-d`	(Required) Module name. Convert template from IPR for the selected modules. Valid options: **`stitch`** or **`register`**.
	`-l`	(Optional, default to 30) Cross limbs length in pixel.
	`-w`	(Optional, default to 1) Cross line thickness in pixel.
	`-i`	(Required) ImageQC/ImageStudio processed staining image TAR.GZ file.
	`-c`	(Required) IPR (image process record) file. IPR is designed to efficiently hold image information generated from each processing step and datasets in various compound types along with metadata. Please check **Stereo-seq File Format Manual** to get more information on IPR file.
	`-d`	(Required) Segmentation module names. Convert segmentation mask TIFF from IPR for the selected modules. Valid options: `tissue` and `cell`. Separate modules by space.
	`-r`	(Required; default to True) Whether output registered images or pre-registered images. “Pre-registered” state stands for the images that have been stitched to a single panoramic image and transformed to have the same scale with the gene expression matrix, but still need to be flipped, rotated, or translated. The options are `True` for output registered images and `False` for output pre-registered images.
	`-o`	(Required) Path to the directory where to store the `imageTools ipr2img` outputs.

2.14.3.2 Usage Examples

Function 1: **img2rpi**

```
$ singularity exec SAW_v7.0.sif imageTools img2rpi \
    -i /path/to/output/04.register/<stainType1>_fov_stitched_transformed.tif,/
path/to/output/04.register/<stainType2>_fov_stitched_transformed.tif \
    -g <stainType1>/<ImageType>,<stainType2>/<ImageType>\
    -b 1 10 50 100 \
    -o /path/to/output/04.register/fov_stitched_transformed.rpi  ## this RPI is
used for manual registration in StereoMap
```

```
# An easy example of transforming multiple <stainType>_fov_stitched_transformed.tif
# and storing as '<stainType>/Image' into fov_stitched_transformed.rpi

$ registerTif=$(find /path/to/output/04.register -maxdepth 1 -name \
*fov_stitched_transformed.tif)
$ regTifStr=$(echo $registerTif | tr ' ' ',')
$ regGroup=$(find /path/to/output/04.register -maxdepth 1 -name \
'*fov_stitched_transformed.tif' -exec sh -c 'for f do basename -- "$f" \
_fov_stitched_transformed.tif;done' sh {} +)
$ regGroupStr=$(echo $regGroup | sed 's/ \|$/\/Image,/g' | sed 's/.$//') \

$ singularity exec SAW_v7.0.sif imageTools img2rpi \
    -i ${regTifStr} \
    -g ${regGroupStr} \ ## <stainType>/Image, <stainType>/Image
    -g 1 10 50 100 \
    -o /path/to/output/04.register/fov_stitched_transformed.rpi
```

* Please be noted that we use the backward slash "\" to indicate the end of a line in a command that spans multiple lines.

```
## Cutomize labeled-tissue-mask color

$ singularity exec SAW_v7.0.sif imageTools img2rpi \
    -i /path/to/<stainType>_{SN}_<label01>_tissue_cut.tif,/path/to/<stainType>_{SN}
_<label02>_tissue_cut.tif\
    -g <stainType>/TissueMask_<label01>,<stainType>/TissueMask_<label02> \
    -o /path/to/output/04.register/<stainType>_stitch_check_tmplt.tif \
    -g 1 10 50 100 \
    -c \(100,200,255\),\(200,25,255\) \  ## escape character "|" before brackets
    -o /path/to/fov_stitched.rpi  ## this RPI could be used for manual registration
in StereoMap
```

Function 2: **`merge` (`-i` Arbitrary up to three grayscale images (.tif))**

```
$ singularity exec SAW_v7.0.sif imageTools merge \
    -i /path/to/output/04.register/<stainType>_{SN}_regist.tif,/path/to/output/04.
register/<stainType>_{SN}_tissue_cut.tif,/path/to/output/04.register/<stainType>_
{SN}_mask.tif \
    -o /path/to/output/04.register/merge.tif
```

* Please be noted that we use the backward slash "\" to indicate the end of a line in a command that spans multiple lines.

Function 3: **`overlay`**

Overlay stitching template in IPR or in a TXT format with the pre-registered panoramic image to check the stitching result.

```
## stitch
preRegImage=/path/to/output/04.register/<stainType>_fov_stitched_transformed.tif
imageIPR=$(find /path/to/output/04.register -maxdepth 1 -name {SN}*.ipr | head -1)
stitchTmplt=/path/to/output/04.register/<stainType>_transform_template.txt

singularity exec SAW_v7.0.sif imageTools overlay \
    -i ${preRegImage} \
    -c ${imageIPR} \
    -o /path/to/output/04.register/<stainType>_stitch_check.tif \
    -d stitch \
    -l 60 \
    -w 3

singularity exec SAW_v7.0.sif imageTools overlay \
    -i ${preRegImage} \
    -c ${stitchTmplt} \
    -o /path/to/output/04.register/<stainType>_stitch_check_tmplt.tif \
    -d stitch \
    -l 60 \
    -w 3
```

Overlay registration template in IPR or in a TXT format with the registered panoramic image.

```
## register
$ regImage=/path/to/output/04.register/<stainType>_{SN}_regist.tif
imageIPR=$(find /path/to/output/04.register -maxdepth 1 -name {SN}*.ipr | head -1)
regTmplt=/path/to/output/04.register/<stainType>_matrix_template.txt

$ singularity exec SAW_v7.0.sif imageTools overlay \
    -i ${regImage} \
    -c ${imageIPR} \
    -o /path/to/output/04.register/register_check.tif \
    -d register \
    -l 60 \
    -w 3

$ singularity exec SAW_v7.0.sif imageTools overlay \
    -i ${regImage} \
    -c ${regTmplt} \
    -o /path/to/output/04.register/register_check_tmplt.tif \
    -d register \
    -l 60 \
    -w 3
```

Function 4: `ipr2img`

Output images (.tif) from IPR file. Other usages of this function can be found in the tutorial for manual processing workflow on GitHub **(https://github.com/STOmics/SAW/blob/main/Documents/UserManual/).**

2.14.4 manualRegister

SAW manualRegister pipeline is performed when automatic registration result does not meet the requirement. Users can use **StereoMap**, an offline visualization software, to manually register images with the gene expression matrix. The operation parameters are recorded in the JSON file and can be input into the SAW `manualRegister` pipeline to modify registration record in IPR. Remember to run `imageTools ipr2img` again to acquire transformed images.

Running `manualRegister` requires the following files and information:

- count output gene expression matrix file **(.raw.gef)**
- An image processing output directory, including several `*fov_stitched_transformed.tif` files, usually in `/path/to/output/04.register`
- `register/rapidRegister` output IPR file **(.ipr)**, supporting v0.1.0
- **StereoMap** manual registration output JSON file, recording manual operations **(.regist.json)**

Expected running time for 1 cm x 1 cm Stereo-seq Chip T image: ~3 min, Memory: 7 G

2.14.4.1 Arguments and Options

Table 2-19 `manualRegister` Arguments and Options

Parameters	Function
`-i`	(Required) `register/rapidRegister` output directory which includes a `<stainType>_fov_stitched_transformed.tif` or `<stainType>_fov_stitched.tif` file.
`-c`	(Required) `register/rapidRegister` output IPR file. `manualRegister` would overwrite registration parameters in the IPR to the manually adjusted parametes acquired from **StereoMap**.
`-v`	(Required) `count` output gene expression matrix GEF file.
`-m`	(Optional; default to "") The path to StereoMap outpu JSON file
`-w`	(Optional; default to False) False: negative degree when clockwise rotation. True: positive degree when clockwise rotation. Being valid when -m is missing.
`-o`	(Optional; default to 0 0) Offset in the x and y direction from the center of the `fov_stitched_transformed.tif` image. These two values are specified as "offsetX" and "offsetY" in the **StereoMap** output JSON file created after manual registration, and will be recorded as "OffsetX" and "OffsetY" in the IPR "Register" group. The offset x and offset y are separate by space.
`-f`	(Optional; default to 0) Whether flip the image horizontally along the y-axis. This value is specified as "flip" in the **StereoMap** output JSON file created after manual registration. Valid options: 1 (flip), 0 (not flip). Flip information will be recorded as "Flip" in the IPR "Register" group.
`-r`	(Optional; default to 0) The manual rotation degree from the center of the fov_stitched_transformed.tif image. This value is specified as "rotate" in the **StereoMap** output JSON file created after manual registration, and will be recorded as "ManualRotation" in the IPR "Register" group.
`-s`	(Optional; default to 0 0) The manual scale in the x and y direction from the center off the `fov_stitched_transformed.tif` image. These two values are specified as "extrude_x" and "extrude_y" in the **StereoMap** output JSON file created after manual registration, and will be recorded as "ManualScaleX" and "ManualScaleY" in the IPR "Register" group. The scale x and scale y are separated by space.
`-p`	(Required) Output directory.

2.14.4.2 Usage Examples

Tip: get the inputs for `manualRegister` from StereoMap manual registration output JSON file:

```
Option 1:
- "offsetX":{offsetX} of -o
- "offsetY":{offsetY} of -o
- "flip":{flip} of -f
- "rotate": {rotate} of -r
- "extrude_x": {extrude_x} of -s
- "extrude_y": {extrude_y} of -s
- "version":{isReversed} of -w,If this key is not present in the JSON, enter "True". If it is present, enter "False". This parameter only takes effect when -m is missing.

Option 2: Use StereoMap output JSON file only by -m.
```

Scenario 1: transform image based solely on manual operation

```
$ mkdir -p /path/to/output/manualregister
$ cp $(find /path/to/output/04.register -maxdepth 1 -name {SN}*.ipr | head -1)
/path/to/output/manualregister  # we recommend to make a copy of IPR to pre-
vent overwirte original file
$ regIPR=$(find /path/to/output/manualregister -maxdepth 1 -name {SN}*.ipr |
head -1)

$ singularity exec SAW_v7.0.sif manualRegister \
    -i /path/to/output/04.register \
    -c ${regIPR} \
    -v /path/to/output/03.count/{SN}.raw.gef \
    -o {offsetX} {offsetY} \
    -f {flip} \
    -r {rotate} \
    -s {extrude_x} {extrude_y} \
    -w {isReversed} \
    -p /path/to/output/manualregister
```

Scenario 2: Use StereoMap output JSON file only

```
$ mkdir -p /path/to/output/manualregister
$ cp $(find /path/to/output/04.register -maxdepth 1 -name {SN}*.ipr | head -1)
/path/to/output/manualregister  # we recommend to make a copy of IPR to pre-
vent overwirte original file
$ regIPR=$(find /path/to/output/manualregister -maxdepth 1 -name {SN}*.ipr |
head -1)

singularity exec SAW_v7.0.sif manualRegister \
    -i /path/to/output/04.register \
    -c ${regIPR} \
    -v /path/to/output/03.count/{SN}.raw.gef \
    -p /path/to/datetime.regist.json
    -p /path/to/output/manualregister
```

2.14.5 lasso

SAW `lasso` pipeline is performed to extract cell or spatial gene expression matrix(ces) of one or multiple manually delineated closed regions acquired from **StereoMap**.

Running **`lasso`** requires the following files and information:

- GEF file that includes **`wholeExp`** group or **`cellCut`** output cell bin GEF file **(.gef or .cellbin.gef)**
- **StereoMap** output coordinates list file **(.geojson)**

Running time and memory are highly dependent on the selected bin size and data volume.

2.14.5.1 Arguments and Options

Table 2-20 `lasso` Arguments and Options

Parameters	Function
`-i`	(Required) GEF file that includes wholeExp group or cellCut output cell bin GEF file (.gef or .cellbin.gef)
`-m`	(Required) StereoMap output coordinates list file (.geojson)
`-n`	(Required) The Stereo-seq Chip T serial number.
`-o`	(Required) Path to the directory where to store the lasso outputs. It has to be an existing path.
`-s`	(Optional for square bin, default=1) GEM Bin size. Separate multiple bin size by comma, e.g. "-s 10,20".

2.14.5.2 Usage Examples

Please Note! Replace `{SN}` with your Stereo-seq Chip T serial number (SN, e.g. SS200000135TL_D1), `{bin}` with the bin size you want (e.g. 1, 10, 200), and `/upload/path/{taskID}` with the true path and task ID.

Scenario 1: square bin lasso

```
$ mkdir -p /path/to/output/lasso

$ singularity exec SAW_v7.0.sif lasso \
    -i /path/to/output/03.count/{SN}.gef \
    -m /upload/path/{taskID}.anno.geojson \
    -o /path/to/output/lasso \
    -s {bin} \
    -n {SN}
```

Scenario 2: cell bin lasso

```
$ mkdir -p /path/to/output/lasso

$ singularity exec SAW_v7.0.sif lasso \
    -i /path/to/output/051.tissuecut/{SN}.cellbin.gef \
    -m /upload/path/{taskID}.anno.geojson \
    -o /path/to/output/lasso \
    -n {SN}

# do cell cluster for cellbin.gef and can show clusters in StereoMap
$ singularity exec SAW_v7.0.sif lasso \
    -i /path/to/output<label>/{SN}.<label>.label.cellbin.gef \
    -o /path/to/output<label>/{SN}.<label>.label.cell.cluster.h5ad

# pre-compute rendering data and can show cellbin.gef expression heatmap in
StereoMap
$ singularity exec SAW_v7.0.sif lasso \
    -i /path/to/output<label>/{SN}.<label>.label.cell.cluster.h5ad \
    -o /path/to/<label>
```

2.14.5.3 Outputs

Square bin `lasso` output files are:

```
$ tree /path/to/output/lasso
/path/to/output/lasso
└── <label>
    ├── SN.<label>.label.gef
    └── segmentation
        ├── SN.lasso.<binList[0]>.<label>.gem.gz
        ...
        ├── SN.lasso.<binList[m]>.<label>.gem.gz
        └── SN.lasso.<label>.mask.tif
```

Cellbin `lasso` output files are:

```
$ tree /path/to/output/lasso
/path/to/output/lasso
└── <label>
    └── SN.<label>.label.cellbin.gef
```

2.14.6 cellChunk

`cellChunk` is performed to generate precomputed rendering data for StereoMap. This information will be recorded in `/codedCellBlock` in cell bin GEF file.

Running **`cellChunk`** requires the following file:

- cell bin GEF file **(cellbin.gef)** after **`cellCluster`**.

2.14.6.1 Arguments and Options

Table 2-21 `cellChunk` Arguments and Options

Parameters	Function
`-i`	(Required) cell bin GEF file (.cellbin.gef or .cellbin.gef) after **`cellCluster`**.
`-o`	(Required) Path to the directory where to store outputs. It has to be an existing path.

2.14.6.2 Usage Examples

```
$ mkdir -p /path/to/output/lasso

$ singularity exec SAW_v7.0.sif cellChunk \
    -i /path/to/output/051.cellcut/{SN}.adjusted.cellbin.gef \
    -o /path/to/051.cellcut
```

CHAPTER 3
TEST DATA DEMONSTRATION

Users may refer to this section as a format for testing SAW process, of the files shown in this chapter as the reference in testing SAW pipelines. This chapter includes the statistics results and examples of critical files for each key step.

SN: SS200000135TL_D1

"..." in the demo stands for some lines of log information that can be omitted.

3.1 mapping

3.1.1 Statistical Report for CID Mapping and Filtering

```
$ cat /path/to/output/01.mapping/E100026571_L01_trim_read_1.CIDMap.stat
...
                      unique_CID_in_mask:        645784920
                        unique_CID_in_fq:        78021796
                             total_reads:        1002214171
                        CID_with_N_reads:        8031                0.00  %
                        CID_mapped_reads:        845113606           84.32 %
                CID_exactly_mapped_reads:        698277934           69.67 %
            CID_mapped_reads_with_mismatch:      146827641           14.65 %
                    discarded_MID_reads:         8451158             0.84  %
                        MID_with_N_reads:        13339               0.00  %
                   MID_with_polyA_reads:         13044               0.00  %
          MID_with_low_quality_base_reads:       8424775             0.84  %
                           reads_with_dnb:       45281336            4.52  %
                    reads_with_adapter:          8136836             0.81  %
          short_reads_filtered_with_polyA:       14191622            1.42  %
                        reads_with_polyA:        105849530           10.56 %
                         reads_with_rRNA:        0                   0.00  %
                        Q10_bases_in_seq:        99.47 %
                        Q20_bases_in_seq:        97.12 %
                        Q30_bases_in_seq:        91.08 %
                        Q10_bases_in_MID:        99.26 %
                        Q20_bases_in_MID:        96.32 %
                        Q30_bases_in_MID:        89.45 %
                        Q10_bases_in_CID:        99.54 %
                        Q20_bases_in_CID:        97.49 %
                        Q30_bases_in_CID:        91.74 %
```

3.1.2 Statistical Report for Reference Genome Alignment

```
$ cat /path/to/output/01.mapping/E100026571_L01_trim_read_1.Log.final.out
...
                          Number of input reads |        769052654
                    Average input read length |        95
                                  UNIQUE READS:
                 Uniquely mapped reads number |        645319822
                      Uniquely mapped reads % |        83.91%
                        Average mapped length |        95.14
                     Number of splices: Total |        67634124
          Number of splices: Annotated (sjdb) |        65711163
                     Number of splices: GT/AG |        66445242
                     Number of splices: GC/AG |        457783
                     Number of splices: AT/AC |        41573
                Number of splices: Non-canonical |        689526
                     Mismatch rate per base, % |        0.50%
                        Deletion rate per base |        0.07%
                      Deletion average length |        3.91
                      Insertion rate per base |        0.03%
                     Insertion average length |        1.25
                           MULTI-MAPPING READS:
       Number of reads mapped to multiple loci |        87828411
            % of reads mapped to multiple loci |        11.42%
       Number of reads mapped to too many loci |        5333051
            % of reads mapped to too many loci |        0.69%
                                UNMAPPED READS:
  Number of reads unmapped: too many mismatches |        0
       % of reads unmapped: too many mismatches |        0.00%
            Number of reads unmapped: too short |        29365488
                 % of reads unmapped: too short |        3.82%
             Number of reads unmapped: other |        1205882
                  % of reads unmapped: other |        0.16%
                                CHIMERIC READS:
                     Number of chimeric reads |        0
                          % of chimeric reads |        0.00%
```

3.1.3 Example of BAM mapping

```
$ samtools view /path/to/output/01.mapping/E100026571_L01_trim_read_1.Aligned.
sortedByCoord.out.bam | head -2
E100026571L1C007R00303973559    256     1       3000644 3       100M    *       0
0       GCCTCATTGTGCCCCATATGTTTGCCTATGTTGTGGACTTATTTTCATTAAACTTTAAAACATCTTTA-
ATTTTTTTCTTTATTTCATCATTGACCAAGCT    -FCA9D?GFFD<-D<CGFEGD-DG*FGFDFBE;E(9BGGE38FFF-
G9GG;0?GGFGB?E@G:GGG3GF79F0GGDG?G<D>F;EG,G?<<CD4>G=>B+C    NH:i:2  HI:i:2  AS:i:94
nM:i:2 Cx:i:8839        Cy:i:7539       UR:Z:120CF
E100026571L1C003R03702347721    0       1       3001778 255     100M    *       0
0       GTATGACATCTGTCCAGGATCTTCTAGCTTTCATAGTCTCTGGTGAGAAGTCTGGAGTAATTCTAATAGG-
CCTGCATTTATATGTTACTTGACCTTTTTC    EEFEDFFEFFFFEFFFFEC@EFFFFDFFEEFFEFFFFCFCEFFAFB-
FCED??FGBEFFDC:FFFDCFAF4FAFFDFFDG?DFBD.F@FECA/FEDEFFAA    NH:i:1  HI:i:1  AS:i:92
nM:i:3 Cx:i:12136       Cy:i:14034      UR:Z:C0808
```

3.2 merge

3.2.1 Example of Mapped CID List with Reads Count File

```
$ head /path/to/output/02.merge/SS200000135TL_D1.merge.barcodeReadsCount.txt
7127    18002   48
4348    19028   1
14130   8635    1
7618    14537   24
4912    10945   5
16783   12914   1
15539   8177    1
9288    8082    14
7274    16533   59
9087    10657   10
```

3.3 count

3.3.1 Statistical Report for MID Filtering and Gene Annotation

```
$ cat /path/to/output/03.count/SS200000135TL_D1.Aligned.sortedByCoord.out.merge.q10.dedup.
target.bam.summary.stat
## FILTER & DEDUPLICATION METRICS
TOTAL_READS     PASS_FILTER     ANNOTATED_READS UNIQUE_READS    FAIL_FILTER_RATE
FAIL_ANNOTATE_RATE      DUPLICATION_RATE
733148233       645319822       533743961       108363128       11.98   17.29   79.70
## ANNOTATION METRICS
TOTAL_READS     MAP             EXONIC          INTRONIC        INTERGENIC      TRANSCRIP-
TOME    ANTISENSE
645319822       645319822       484480416       49263545        111575861       533743961
110185975
100.0           100.0           75.1            7.6             17.3            82.7
17.1
```

3.3.2 Example of Annotated BAM

```
$ samtools view /path/to/output/03.count/SS200000135TL_D1.Aligned.sortedByCoord.out.merge.
q10.dedup.target.bam | head -2
E100026571L1C007R00303973559    768     1       3000644 3       100M    *       0       0
GCCTCATTGTGCCCCATATGTTTGCCTATGTTGTGGACTTATTTTCATTAAACTTTAAAACATCTTTAATTTTTTTCTTTATTTCAT-
CATTGACCAAGCT    -FCA9D?GFFD<-D<CGFEGD-DG*FGFDFBE;E(9BGGE38FFFG9GG;0?GGFGB?E@G:GGG3GF79F-
0GGDG?G<D>F;EG,G?<<CD4>G=>B+C    NH:i:2  HI:i:1  AS:i:94 nM:i:2  Cx:i:8839       Cy:i:7539
UR:Z:120CF
E100026571L1C003R03702347721    0       1       3001778 255     100M    *       0       0
GTATGACATCTGTCCAGGATCTTCTAGCTTTCATAGTCTCTGGTGAGAAGTCTGGAGTAATTCTAATAGGCCTGCATTTATAT-
GTTACTTGACCTTTTTC    EEFEDFFEFFFFEFFFFEC@EFFFFDFFEEFFEFFFFCFCEFFAFBFCED??FGBEFFDC:FFF-
DCFAF4FAFFDFFDG?DFBD.F@FECA/FEDEFFAA    NH:i:1  HI:i:1  AS:i:92 nM:i:3  Cx:i:12136
Cy:i:14034      UR:Z:C0808      XF:i:2
```

3.3.3 Example of count Gene Expression Matrix

```
$ h5dump -n /path/to/output/03.count/SS200000135TL_D1.raw.gef
HDF5 “/path/to/output/03.count/SS200000135TL_D1.raw.gef” {
FILE_CONTENTS {
 group      /
 group      /geneExp
 group      /geneExp/bin1
 dataset    /geneExp/bin1/exon
 dataset    /geneExp/bin1/expression
 dataset    /geneExp/bin1/gene
 }
}

$ h5dump -d /geneExp/bin1/expression  /path/to/output/03.count/SS200000135TL_
D1.raw.gef | head -15
HDF5 “/path/to/output/03.count/SS200000135TL_D1.raw.gef” {
DATASET “/geneExp/bin1/expression” {
   DATATYPE  H5T_COMPOUND {
      H5T_STD_U32LE “x”;
      H5T_STD_U32LE “y”;
      H5T_STD_U8LE “count”;
   }
   DATASPACE  SIMPLE { ( 76210618 ) / ( 76210618 ) }
   DATA {
   (0): {
         636,
         12671,
         2
      },
   (1): {

$ h5dump -d /geneExp/bin1/gene /path/to/output/03.count/SS200000135TL_D1.raw.gef |
head -20
HDF5 “/path/to/output/03.count/SS200000135TL_D1.raw.gef” {
DATASET “/geneExp/bin1/gene” {
   DATATYPE  H5T_COMPOUND {
      H5T_STRING {
         STRSIZE 32;
         STRPAD H5T_STR_NULLTERM;
         CSET H5T_CSET_ASCII;
         CTYPE H5T_C_S1;
      } “gene”;
      H5T_STD_U32LE “offset”;
      H5T_STD_U32LE “count”;
   }
   DATASPACE  SIMPLE { ( 24670 ) / ( 24670 ) }
   DATA {
   (0): {
         “0610005C13Rik”,
         0,
         45
      },
   (1): {
```

3.3.4 Example of count Sampling File

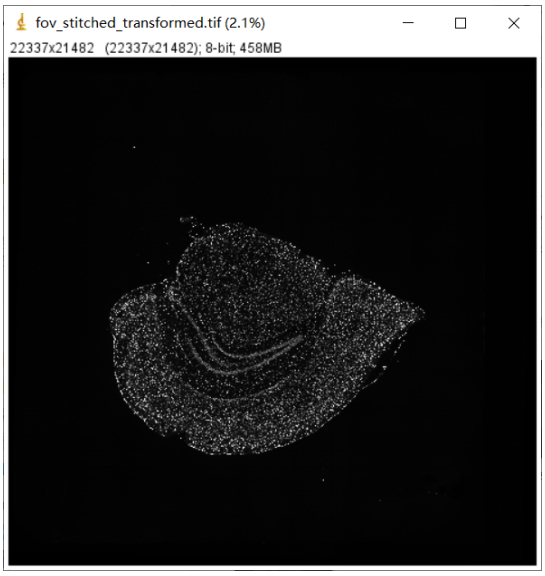
```
$ head -8 /path/to/output/03.count/SS200000135TL_D1_raw_barcode_gene_exp.txt
x y geneIndex MIDIndex readCount
9602 7705 10551 611723 2
4888 10392 10551 665954 4
8901 7096 10551 881671 1
8901 7096 10551 357383 20
7397 18783 10551 355789 1
9155 13032 10551 297666 1
9155 13032 10551 298690 1
```

3.4 register and imageTools

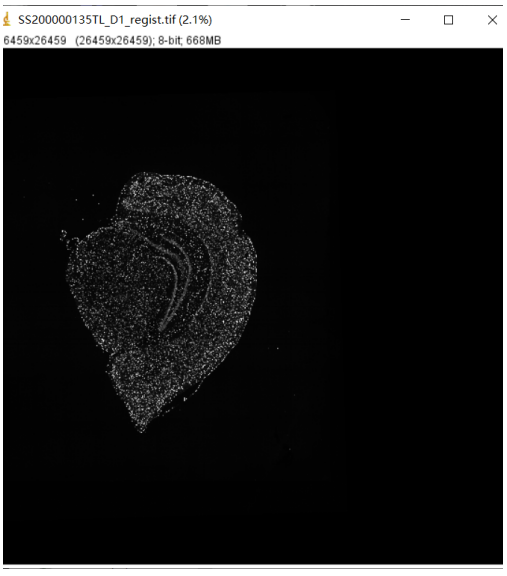
3.4.1 Registered image

File /path/to/output/04.register/ssDNA_fov_stitched_transformed.tif and /path/to/output/04.register/ssDNA_SS200000135TL_D1_regist.tif.

/path/to/output/04.register/ssDNA_fov_stitched_transformed.tif



/path/to/output/04.register/ssDNA_SS200000135TL_D1_regist.tif

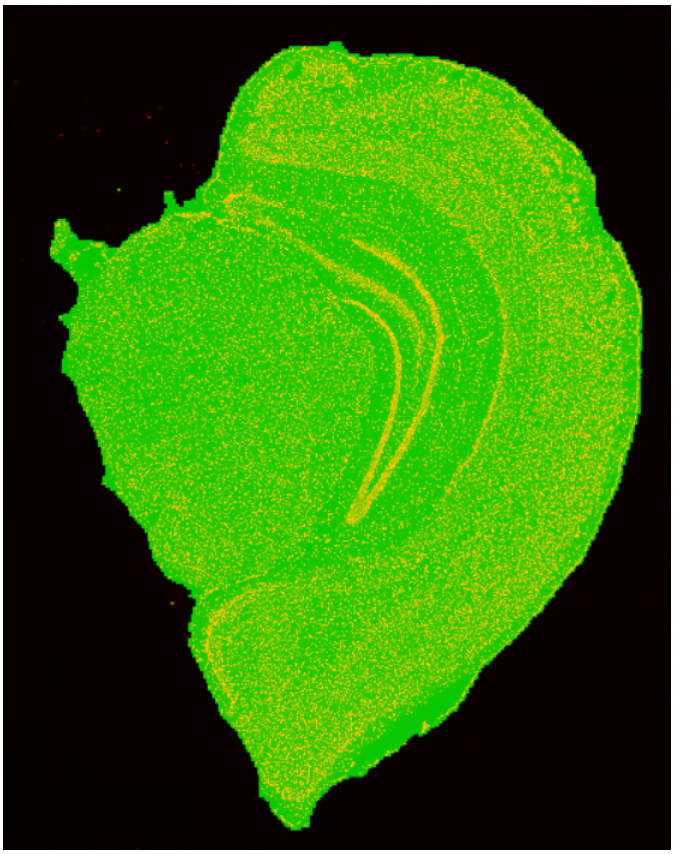


3.4.2 Image Process Record File

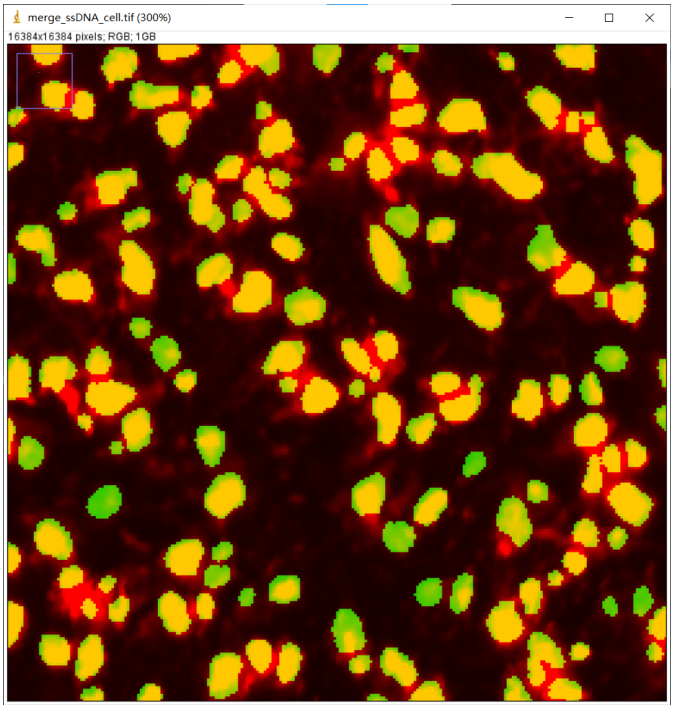
```
$ h5dump -n /path/to/output/04.register/SS200000135TL_D1_20220527_201353_1.1.0.ipr
HDF5 "/path/to/output/04.register/SS200000135TL_D1_20220527_201353_1.1.0.ipr" {
FILE_CONTENTS {
 group      /
 group      /ManualState
 dataset    /Preview
 group      /StereoResepSwitch
 group      /ssDNA
 group      /ssDNA/CellSeg
 dataset    /ssDNA/CellSeg/CellMask
 dataset    /ssDNA/CellSeg/CellSegTrace
 group      /ssDNA/ImageInfo
 dataset    /ssDNA/ImageInfo/RGBScale
 group      /ssDNA/QCInfo
 group      /ssDNA/QCInfo/CrossPoints
 dataset    /ssDNA/QCInfo/CrossPoints/0_0
 ...
 dataset    /ssDNA/QCInfo/CrossPoints/9_8
 group      /ssDNA/Register
 dataset    /ssDNA/Register/MatrixTemplate
 group      /ssDNA/Stitch
 group      /ssDNA/Stitch/ScopeStitch
 dataset    /ssDNA/Stitch/ScopeStitch/GlobalLoc
 dataset    /ssDNA/Stitch/ScopeStitch/ScopeHorizontalJitter
 dataset    /ssDNA/Stitch/ScopeStitch/ScopeJitterDiff
 dataset    /ssDNA/Stitch/ScopeStitch/ScopeVerticalJitter
 group      /ssDNA/Stitch/StitchEval
 dataset    /ssDNA/Stitch/StitchEval/GlobalDeviation
 dataset    /ssDNA/Stitch/StitchEval/StitchEvalH
 dataset    /ssDNA/Stitch/StitchEval/StitchEvalV
 dataset    /ssDNA/Stitch/TemplatePoint
 dataset    /ssDNA/Stitch/TransformTemplate
 group      /ssDNA/TissueSeg
 dataset    /ssDNA/TissueSeg/TissueMask
 }
}

$ h5dump -A /path/to/output/04.register/SS200000135TL_D1_20220527_201353_1.1.0.ipr
| head -20
HDF5 “/path/to/output/04.register/SS200000135TL_D1_20220527_201353_1.1.0.ipr” {
GROUP "/" {
   ATTRIBUTE "IPRVersion" {
      DATATYPE  H5T_STRING {
         STRSIZE H5T_VARIABLE;
         STRPAD H5T_STR_NULLTERM;
         CSET H5T_CSET_UTF8;
         CTYPE H5T_C_S1;
      }
      DATASPACE  SCALAR
      DATA {
      (0): "0.2.0"
      }
   }
   GROUP "ManualState" {
      ATTRIBUTE "cellseg" {
         DATATYPE  H5T_ENUM {
            H5T_STD_I8LE;
            "FALSE"            0;
            "TRUE"             1;
```

3.4.3 ImageTools merge

Merged image of microscopy image `ssDNA_SS200000135TL_D1_regist.tif` and tissue segmentation mask file `ssDNA_SS200000135TL_D1_tissue_cut.tif` to check tissue segmentation performance.

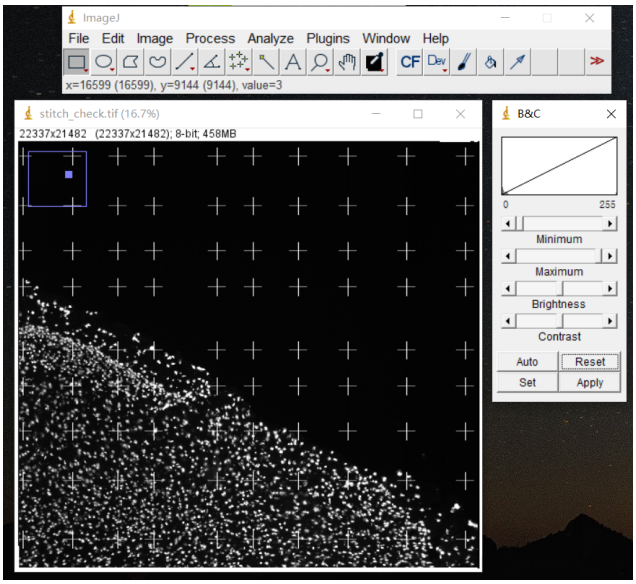


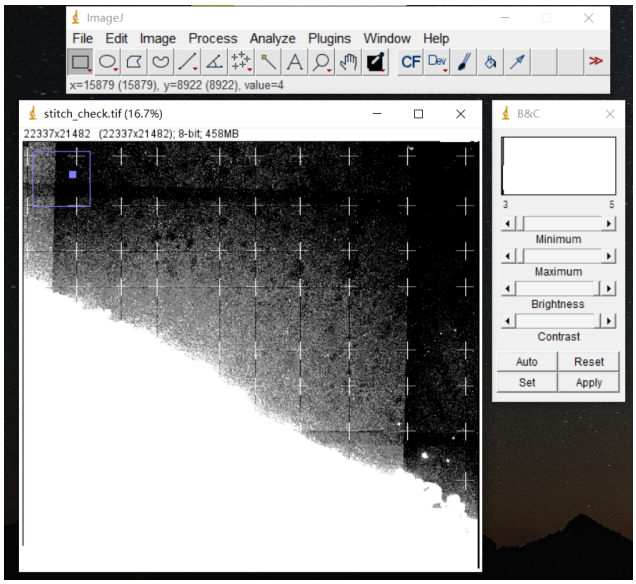
Part of the merged image of microscopy image `ssDNA_SS200000135TL_D1_regist.tif` and cell segmentation mask file `ssDNA_SS200000135TL_D1_mask.tif` to check cell segmentation performance.



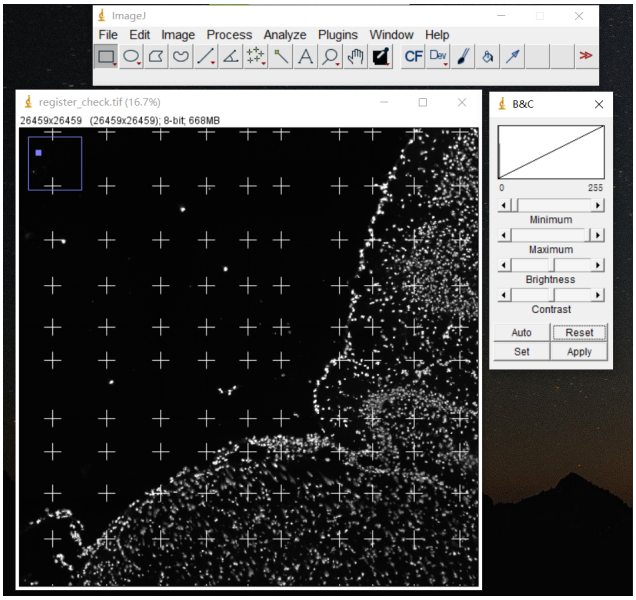
3.4.4 ImageTools overlay

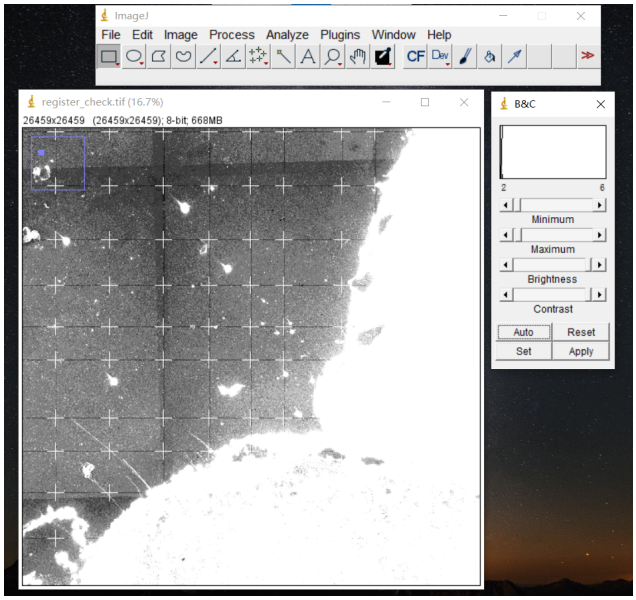
Stack stitching template onto the `ssDNA_fov_stitched_transformed.tif` to check the result of stitching.





Stack registration template onto the `ssDNA_SS200000135TL_D1_register.tif` to check the result of registration.





3.5 tissueCut

3.5.1 Statistical Report for Tissue Covered Region

```
$ cat /path/to/output/05.tissuecut/tissuecut.stat
# Tissue Statistic Analysis with Stain Image
Contour_area: 87086375
Number_of_DNB_under_tissue: 36521212
Ratio: 41.94%
Total_gene_type: 24289
MID_counts: 89679129
Fraction_MID_in_spots_under_tissue: 82.76%
Reads_under_tissue: 709807297
Fraction_reads_in_spots_under_tissue: 83.99%

binSize=1
Mean_reads_per_spot: 15.10
Median_reads_per_spot: 9.00
Mean_gene_type_per_spot: 1.71
Median_gene_type_per_spot: 1
Mean_Umi_per_spot: 2.46
Median_Umi_per_spot: 2

binSize=20
Mean_reads_per_spot: 3241.11
Median_reads_per_spot: 2782.00
Mean_gene_type_per_spot: 241.08
Median_gene_type_per_spot: 227
Mean_Umi_per_spot: 409.56
Median_Umi_per_spot: 370

binSize=50
Mean_reads_per_spot: 20054.45
Median_reads_per_spot: 18285.00
Mean_gene_type_per_spot: 1165.56
Median_gene_type_per_spot: 1133
Mean_Umi_per_spot: 2534.10
Median_Umi_per_spot: 2346

binSize=100
Mean_reads_per_spot: 78867.48
Median_reads_per_spot: 72545.00
Mean_gene_type_per_spot: 3110.83
Median_gene_type_per_spot: 3117
Mean_Umi_per_spot: 9964.35
Median_Umi_per_spot: 9205

binSize=150
Mean_reads_per_spot: 174614.34
Median_reads_per_spot: 162073.00
Mean_gene_type_per_spot: 4926.51
Median_gene_type_per_spot: 5065
Mean_Umi_per_spot: 22066.71
Median_Umi_per_spot: 20430

binSize=200
Mean_reads_per_spot: 305687.91
Median_reads_per_spot: 285723.00
Mean_gene_type_per_spot: 6424.38
Median_gene_type_per_spot: 6747
Mean_Umi_per_spot: 38621.50
Median_Umi_per_spot: 36060
```

3.5.2 Example of Gene Expression Matrix for Tissue Covered Region

```
$ h5dump -n /path/to/output/05.tissuecut/SS200000135TL_D1.tissue.gef
HDF5 "/path/to/output/05.tissuecut/SS200000135TL_D1.tissue.gef" {
FILE_CONTENTS {
 group      /
 group      /geneExp
 group      /geneExp/bin1
 dataset    /geneExp/bin1/exon
 dataset    /geneExp/bin1/expression
 dataset    /geneExp/bin1/gene
 }
}

$ h5dump -d /geneExp/bin1/expression /path/to/output/05.tissuecut/SS200000135TL_D1.
tissue.gef | head -15
HDF5 "/path/to/output/05.tissuecut/SS200000135TL_D1.tissue.gef" {
DATASET "/geneExp/bin1/expression" {
   DATATYPE  H5T_COMPOUND {
      H5T_STD_U32LE "x";
      H5T_STD_U32LE "y";
      H5T_STD_U8LE "count";
   }
   DATASPACE  SIMPLE { ( 62542665 ) / ( 62542665 ) }
   DATA {
   (0): {
         6148,
         10906,
         1
      },
   (1): {

$ h5dump -d /geneExp/bin1/gene /path/to/output/05.tissuecut/SS200000135TL_D1.tis-
sue.gef | head -20
HDF5 "/path/to/output/05.tissuecut/SS200000135TL_D1.tissue.gef" {
DATASET "/geneExp/bin1/gene" {
   DATATYPE  H5T_COMPOUND {
      H5T_STRING {
         STRSIZE 64;
         STRPAD H5T_STR_NULLTERM;
         CSET H5T_CSET_ASCII;
         CTYPE H5T_C_S1;
      } "gene";
      H5T_STD_U32LE "offset";
      H5T_STD_U32LE "count";
   }
   DATASPACE  SIMPLE { ( 24289 ) / ( 24289 ) }
   DATA {
   (0): {
        "0610005C13Rik",
         0,
         24
      },
   (1): {
```

3.5.3 Example of Gene Expression Matrix for complete GEF

```
$ h5dump -n /path/to/output/02.count/SS200000135TL_D1.gef
HDF5 “/path/to/output/05.tissuecut/SS200000135TL_D1.gef” {
FILE_CONTENTS {
 group      /
 group      /geneExp
 group      /geneExp/bin1
 dataset    /geneExp/bin1/exon
 dataset    /geneExp/bin1/expression
 dataset    /geneExp/bin1/gene
 group      /geneExp/bin10
 dataset    /geneExp/bin10/exon
 dataset    /geneExp/bin10/expression
 dataset    /geneExp/bin10/gene
 group      /geneExp/bin100
 dataset    /geneExp/bin100/exon
 dataset    /geneExp/bin100/expression
 dataset    /geneExp/bin100/gene
 group      /geneExp/bin20
 dataset    /geneExp/bin20/exon
 dataset    /geneExp/bin20/expression
 dataset    /geneExp/bin20/gene
 group      /geneExp/bin200
 dataset    /geneExp/bin200/exon
 dataset    /geneExp/bin200/expression
 dataset    /geneExp/bin200/gene
 group      /geneExp/bin50
 dataset    /geneExp/bin50/exon
 dataset    /geneExp/bin50/expression
 dataset    /geneExp/bin50/gene
 group      /geneExp/bin500
 dataset    /geneExp/bin500/exon
 dataset    /geneExp/bin500/expression
 dataset    /geneExp/bin500/gene
 group      /stat
 dataset    /stat/gene
 group      /wholeExp
 dataset    /wholeExp/bin1
 dataset    /wholeExp/bin10
 dataset    /wholeExp/bin100
 dataset    /wholeExp/bin20
 dataset    /wholeExp/bin200
 dataset    /wholeExp/bin50
 dataset    /wholeExp/bin500
 group      /wholeExpExon
 dataset    /wholeExpExon/bin1
 dataset    /wholeExpExon/bin10
 dataset    /wholeExpExon/bin100
 dataset    /wholeExpExon/bin20
 dataset    /wholeExpExon/bin200
 dataset    /wholeExpExon/bin50
 dataset    /wholeExpExon/bin500
 }
}
```

```
$ h5dump -d /stat/gene /path/to/output/02.count/SS200000135TL_D1.gef | head -20
HDF5 "/path/to/output/05.tissuecut/SS200000135TL_D1.gef" {
DATASET "/stat/gene" {
   DATATYPE  H5T_COMPOUND {
      H5T_STRING {
         STRSIZE 64;
         STRPAD H5T_STR_NULLTERM;
         CSET H5T_CSET_ASCII;
         CTYPE H5T_C_S1;
      } "gene";
      H5T_STD_U32LE "MIDcount";
      H5T_IEEE_F32LE "E10";
   }
   DATASPACE  SIMPLE { ( 24670 ) / ( 24670 ) }
   DATA {
   (0): {
         "Gm42418",
         5860952,
         60.1028
      },
   (1): {
```

3.6 cellCut

3.6.1 Example of Gene Expression Matrix for Cell Bins

```
$ h5dump -n /path/to/output/051.cellcut/SS200000135TL_D1.cellbin.gef
HDF5 “/path/to/output/051.cellcut/SS200000135TL_D1.cellbin.gef” {
FILE_CONTENTS {
 group      /
 group      /cellBin
 dataset    /cellBin/blockIndex
 dataset    /cellBin/blockSize
 dataset    /cellBin/cell
 dataset    /cellBin/cellBorder
 dataset    /cellBin/cellExon
 dataset    /cellBin/cellExp
 dataset    /cellBin/cellExpExon
 dataset    /cellBin/cellTypeList
 dataset    /cellBin/gene
 dataset    /cellBin/geneExon
 dataset    /cellBin/geneExp
 dataset    /cellBin/geneExpExon
 }
}
```

3.7 saturation

3.7.1 Example of Sequence Saturation File

```
$ cat /path/to/output/07.saturation/sequence_saturation.tsv
sample  bar_x   bar_y1  bar_y2  bar_umi bin_x   bin_y1  bin_y2  bin_umi
0.05    26687198        0.250475        1       20002730        26687198
0.273744        3030    7041
0.1     53374396        0.389862        1       32565765        53374396
0.409615        4045    11464
0.2     106748792       0.542456        1       48842313        106748792
0.557064        4962    17194
0.3     160123200       0.625182        1       60017028        160123200
0.636707        5435    21128
0.4     213497584       0.677501        1       68852648        213497584
0.6871          5778    24238
0.5     266871984       0.713814        1       76374913        266871984
0.722081        6052    26886
0.6     320246400       0.740741        1       83026752        320246400
0.748035        6234    29228
0.7     373620768       0.761599        1       89071589        373620768
0.768155        6381    31356
0.8     426995168       0.778242        1       94689701        426995168
0.784222        6528    33334
0.9     480369568       0.79188         1       99974373        480369568
0.797395        6639    35194
1       533743961       0.803326        1       104973406       533743961
0.808462        6718    36641
```

3.8 report

3.8.1 Example of Statistical Summary Report

```
$ head /path/to/output/08.report/SS200000135TL_D1.statistics.json
{
    "version": "version_v2",
    "1.Filter_and_Map": {
        "1.1.Adapter_Filter": [
            {
                "Sample_id": "E100026571_L01_trim_read_1",
                "getCIDPositionMap_uniqCIDTypes": "645784920",
                "total_reads": "1002214171",
                "CID_withN_reads": "8031        (0.00  %)",
                "mapped_reads": "845113606   (84.32 %)",
```

3.8.2 HTML Report

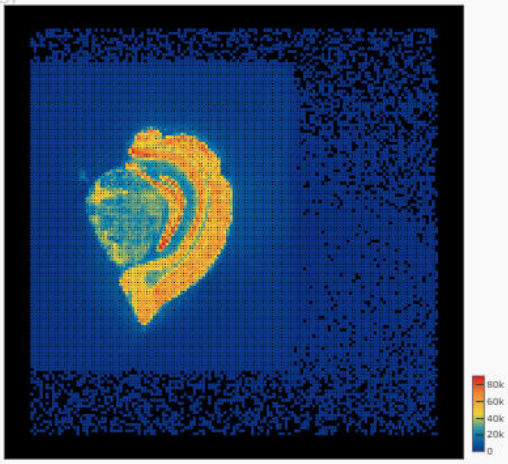
时空组学 STOmics Analytical Report

SS200000135TL_D1

Alerts

Summary Square Bin Cell Bin Image

Spatial Gene Expression Distribution ssDNA



show image — show expression

Tissue Plot with Spot (bin200) Colored by MID Count

1,002,214,171
Total Reads

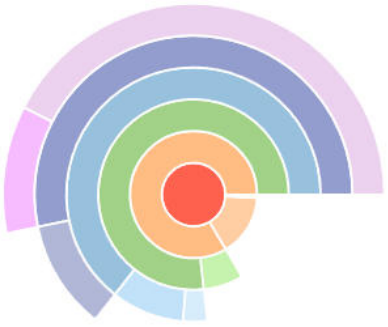
38,621
Mean MID per Bin200

6,424
Mean Gene Type per Bin200

108,363,128
Unique Reads

Key Metrices

Total Reads	1,002,214,171	100%
Valid CID Reads	836,662,448	83.5%
Clean Reads	769,052,654	91.9%
Uniquely Mapped Reads	645,319,822	83.9%
Transcriptome	533,743,961	82.7%
Unique Reads	108,363,128	20.3%
Sequencing Saturation	425,380,833	79.7%
Unannotated Reads	111,575,861	17.3%
Multi-Mapped Reads	93,161,462	12.1%
Unmapped Reads	30,571,370	4.0%
Non-Relevant Short Reads	67,609,794	8.1%
Invalid CID Reads	157,100,565	15.7%
Discarded MID Reads	8,451,158	0.8%



Information

Species	mouse	Tissue	brain
Reference	mouse		
FASTQ	E100026571_L01_trim_read_1		

Sequencing

Total Reads	1,002,214,171
Total Q30	91.08%
CID Q30	91.74%
MID Q30	89.45%

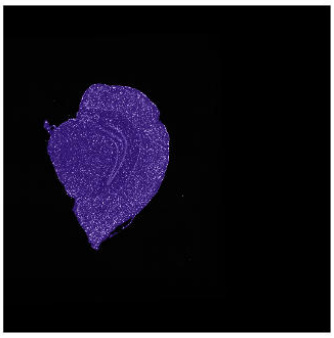
RNA Mapping

Clean Reads	769,052,654
Uniquely Mapped Reads	645,319,822
Multi-Mapped Reads	93,161,462
Unmapped Reads	30,571,370

Annotation

Exonic	484,480,416
Intronic	49,263,545
Intergenic	111,575,861
Transcriptome	533,743,961
Antisense	110,185,975

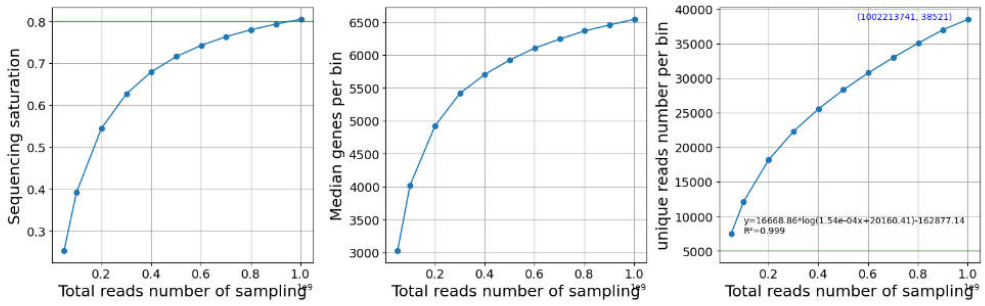
Tissue Segmentation



Tissue

DNB Under Tissue	87,086,375
mRNA-Captured DNBs Under Tissue	36,521,212
Genes Under Tissue	24,289
Number of MID Under Tissue Coverage	89,679,129
Fraction MID in Spots Under Tissue	82.76%
Reads Under Tissue	709,807,297
Fraction Reads in Spots Under Tissue	83.99%

Sequencing Saturation



时空组学 STOmics Analytical Report

SS200000135TL_D1

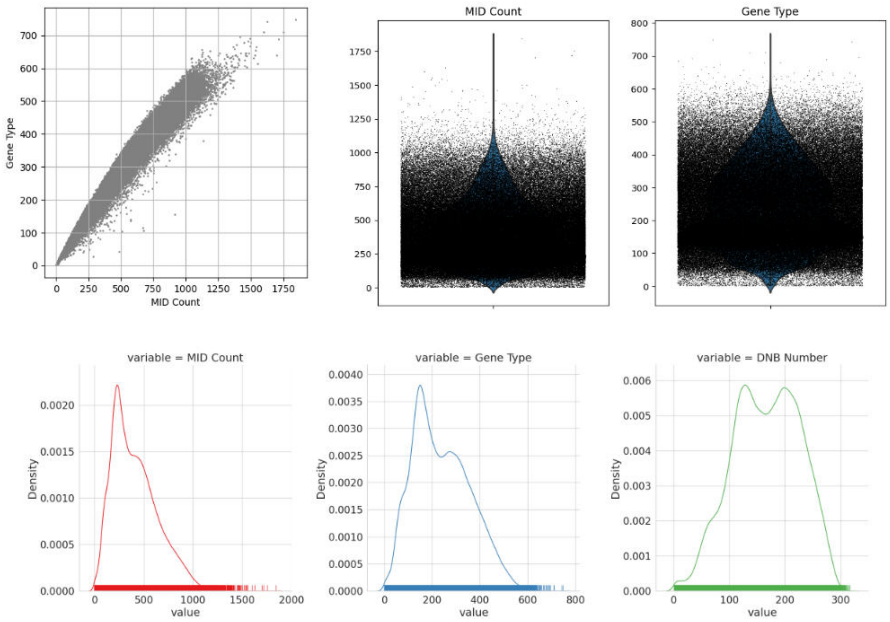
Alerts

Summary Square Bin Cell Bin Image

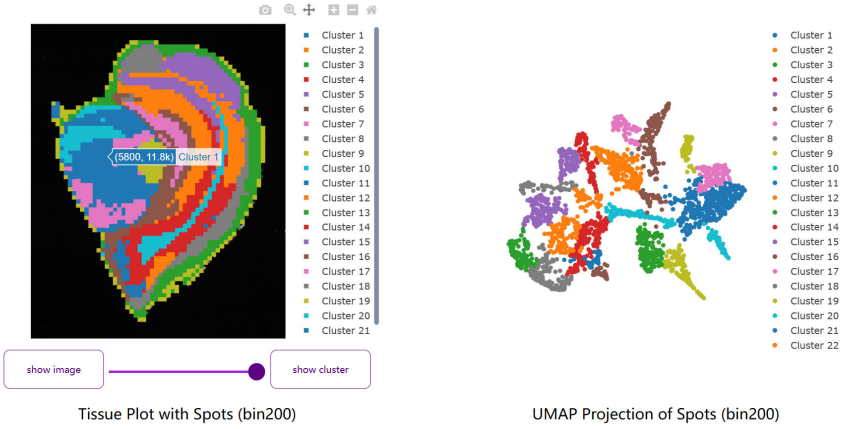
Tissue Square Bin Statistics

Bin Size	Mean Reads (per bin)	Median Reads (per bin)	Mean Gene Type (per bin)	Median Gene Type (per bin)	Mean MID (per bin)	Median MID (per bin)
1	15.10	9	1.71	1	2.46	2
20	3,241	2,782	241	227	409	370
50	20,054	18,285	1,165	1,133	2,534	2,346
100	78,867	72,545	3,110	3,117	9,964	9,205
150	174,614	162,073	4,926	5,065	22,066	20,430
200	305,687	285,723	6,424	6,747	38,621	36,060

Bin 20



Clustering



Tissue Plot with Spots (bin200)

UMAP Projection of Spots (bin200)

时空组学 STOmics Analytical Report

SS200000135TL_D1

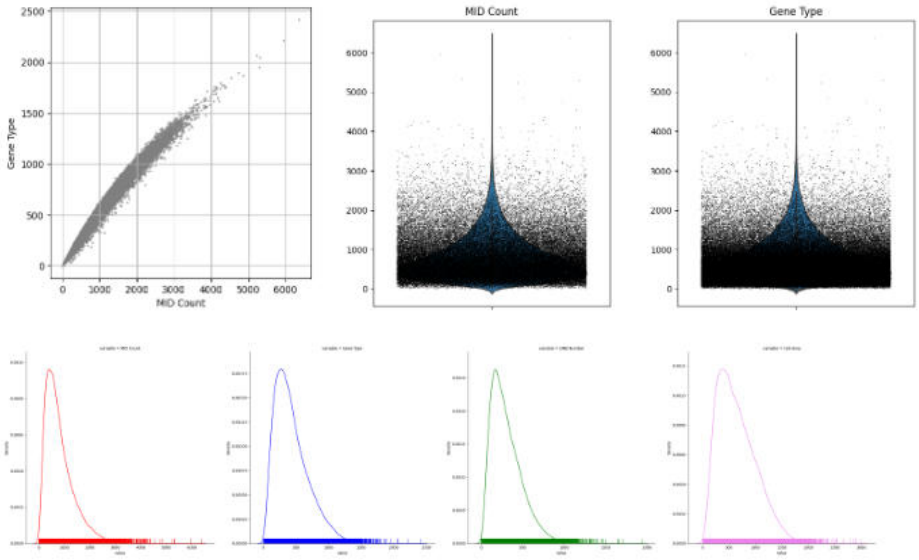
Alerts

Summary | Square Bin | Cell Bin | Image

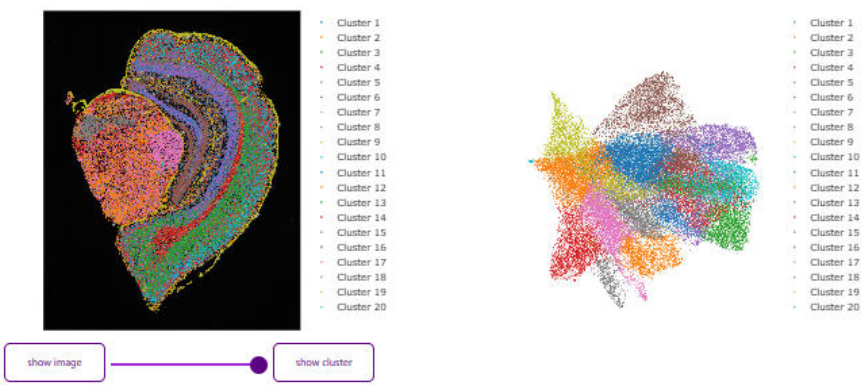
Cell Bin Statistics

Cell Count	76,873		
Mean Cell Area	648	Median Cell Area	586
Mean DNB Count	305	Median DNB Count	268
Mean Gene Type	431	Median Gene Type	378
Mean MID	803	Median MID	666

Cell Bin



Clustering



Tissue Plot with Spots (cell bin)

UMAP Projection of Spots (cellbin)

时空组学 STOmics Analytical Report

SS200000135TL_D1

Alerts

Summary Square Bin Cell Bin Image

Image Information

Manufacture	motic
Model	MoticamS6 M
Scan Time	2021-11-25 12:26:19
Overlap	0.1
Exposure Time (ms)	19.009
Scan Rows	13
Scan Columns	9
FOV Height	2,039
FOV Width	3,064
Stain Type	ssDNA
Stitched Image	No
Image Name	SS200000135TL_D1_SC_20231031_203934_3.0.0

QC

ImageQC Version	3.0.0
QC Pass	Pass
Track Line Score	94
Clarity Score	82
Good FOV Count	98
Total FOV Count	117
Stitching Score	100
Tissue Segmentation Score	100
Registration Score	99

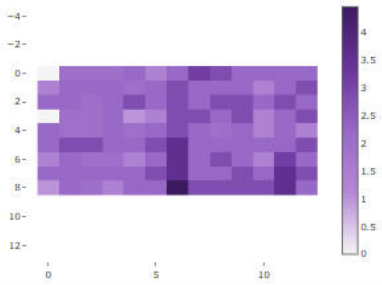
Stitching

Template Source Row No.	7
Template Source Column No.	7
Global Height	24,071
Global Width	25,128

Registration

ScaleX	1.156
ScaleY	1.153
Rotation	-1.564
Flip	True
Image X Offset	-3,367.76
Image Y Offset	2,081
Manual ScaleX	-
Manual ScaleY	-
Manual Rotation	-
Counter Clockwise Rotation	90
Matrix X Offset	0.00
Matrix Y Offset	0.00
Matrix Height	26,459
Matrix Width	26,459

Horizontal Image Stitching Deviation Heatmap



Vertical Image Stitching Deviation Heatmap

Appendices

Appendix A: Recommended Directory Structure for Raw Data

Recommend to organize raw data as below if users prefer to use **SAW shell script** provided on the SAW GitHub page (https://github.com/STOmics/SAW).

```
$ tree
.
|-- image
|   |-- SS200000135TL_D1_SC_20230822_144400_3.0.0.ipr
|   `-- SS200000135TL_D1_SC_20230822_144400_3.0.0.tar.gz
|-- mask
|   `-- SS200000135TL_D1.barcodeToPos.h5
|-- md5
|-- reads
|   |-- E100026571_L01_trim_read_1.fq.gz
|   `-- E100026571_L01_trim_read_2.fq.gz
`-- reference
    |-- STAR_SJ100
    |   |-- chrLength.txt
    |   |-- chrNameLength.txt
    |   |-- chrName.txt
    |   |-- chrStart.txt
    |   |-- exonGeTrInfo.tab
    |   |-- exonInfo.tab
    |   |-- FMindex
    |   |-- geneInfo.tab
    |   |-- Genome
    |   |-- genomeParameters.txt
    |   |-- SA
    |   |-- SAindex
    |   |-- SAindexAux
    |   |-- sjdbInfo.txt
    |   |-- sjdbList.fromGTF.out.tab
    |   |-- sjdbList.out.tab
    |   `-- transcriptInfo.tab
    |-- genes.gtf
    `-- genome.fa

5 directories, 25 files
```

Appendix B: SAW ST Output File List

STEP	OUTPUT	FILE DESCRIPTION
splitMask	*.SN.barcodeToPos.bin	Split Stereo-seq Chip T mask file according to the CID.
mapping	lane.Aligned.sortedByCoord.out.bam	Binary Alignment/Map file, used for storing the sequence alignment information.
	lane.barcodeReadsCount.txt	Mapped CID list file with reads counts for each CID, the three columns record x, y, and read count.
	lane.Log.final.out	Summary mapping statistics after mapping job is complete (STAR output)
	lane.Log.out	Main log file in STAR mapping (STAR output)
	lane.Log.progress.out	Reports job process statistics (STAR output)
	lane.SJ.out.tab	Splice junctions detected in the mapping (STAR output)
	lane.bcPara	A parameters file defines CID mapping options
	lane.CIDMap.stat	Statistical report of CID mapping, such as CID mapped reads count, reads sequencing quality, mapped DNB count, etc.
	lane.run.log	Log file of mapping module.
	lane.valid_cid_reads.fq	**Valid CID Reads** in FASTQ format after CID mapping. Similar to SE FASTQ format but with different first row of each read, for instance, "@V350044321L1C001R0020993658|Cx:i:10413|Cy:i:7737 D3450E0D391E EC7FF", where Cx and Cy represent the decoded coordinates and are added after readID.
	lane.unmapped_reads.fq	Un-mapped reads in FASTQ format after mapping **Clean Reads** to the reference genome. Similar to SE FASTQ format but with different first row of each read, for instance, "@V350044321L1C001R0020993658|Cx:i:10413|Cy:i:7737 D3450E0D391E EC7FF", where Cx and Cy represent the decoded coordinates and are added after readID.
merge	SN.merge.barcodeReadsCount.txt	Mapped CID list file with reads counts for each CID., the three columns record x, y, and read count.
count	SN.Aligned.sortedByCoord.out.merge.q10.dedup.target.bam	Annotated BAM file sorted by coordinate, includes uniquely mapped reads and multi-mapped reads whose HI:i tag is 1.
	SN.Aligned.sortedByCoord.out.merge.q10.dedup.target.bam.csi	Index file of annotated BAM.
	SN.Aligned.sortedByCoord.out.merge.q10.dedup.target.bam.summary.stat	Statistical summary report file for count. Total reads in filter&deduplication metrics stand for the total alignment record count in BAM. Pass filter and total reads in annotation metrics are the same, they represent uniquely mapped reads that will be used for annotation, MID correction and quantification.
	SN.raw.gef	Gene expression file in hdf5 format. This is the first raw matrix that includes the expression information over a complete chip region. It only includes geneExp group for the bin size of 1. The origin of expression matrix has been calibrated to (0,0), and the offset x and y has been record as minX and minY in the attribute of geneExp/expression dataset.
	SN_raw_barcode_gene_exp.txt	A space-separate file records coordinate, gene, MID, and count information. Which is prepared to be a sampling file that performs sequence saturation. The 5 columns are y, x, geneIndex, MIDIndex, readCount.
	count_data_hhmmss.log	Log file.

STEP	OUTPUT	FILE DESCRIPTION
register&imageTools ipr2img	date-hh-mm-ss.log	Log file.
	SN or other user specified name for the image folder used when input into ImageQC/ImageStudio	This subdirectory stores raw small image pieces in TIFF format.
	SN_0000_0000_YYYY-MM-DD_hh-mm-ss-n.tif	Small image pieces in TIFF format.
	<stainType>_fov_stitched_transformed.rpi	Multiple staining stitched panaromic images (.tif) are stored in RPI file. The stitched panoramic image that has pre-registered with the track line pattern template. So that it will not need to further rotate a non-right angle or scale.
	fov_stitched.rpi	Multiple staining stitched panaromic images (.tif) are stored in RPI file. The stitched panoramic image. It needs to further rotate a non-right angle or scale.
	<stainType>_fov_stitched_transformed.tif	The stitched panoramic image from DAPI/IF layers that has pre-registered with the track line pattern template. So that it will not need to further rotate a non-right angle or scale.
	<stainType>_fov_stitched.tif	The stitched panoramic image. It needs to further rotate a non-right angle or scale.
	<stainType>matrix_template.txt	Track line cross point template for the registered staining image. Used for assessing registeration result.
	SN_<chipType>date_time_version.ipr	IPR format image process record file which records basic image information gathered from ImageQC/ImageStudio.
	<stainType>_SN_mask.tif	Registered cell segmentation binary image file from staining layers in TIFF format.
	<stainType>_SN_regist.tif	Registered panoramic image file from staining layers in TIFF format.
	SN.rpi	Registered panoramic image file, tissue area boundary, and cell boundaries (downsampled) that stores as an image pyramid.
	<stainType>_SN_tissue_cut.tif	The tissue segmentation result of registered panoramic image file from staining layers in TIFF format.
	<stainType>_transform_template.txt	Track line cross point template for the <stainType>_fov_stitched_transformed.tif. Used for assessing stitching result.
tissueCut	SN.gef	A gene expression file in hdf5 format. This file is a complete GEF format which includes geneExp group and wholeExp group in bin1, 10, 20, 50, 100, 200, and 500. It also includes a stat group. The original expression matrix has been calibrated to (0,0), and the offset x and y has been recorded as minX and minY in the attribute of geneExp/expression dataset.
	SN.tissue.gef	A gene expression file in hdf5 format. Tissue GEF includes the expression information of the tissue coverage region. It only includes geneExp group for the bin size of 1. The origin of expression matrix has been calibrated to (0,0), and the offset x and y has been record as minX and minY in the attribute of geneExp/expression dataset same to raw GEF.

STEP	OUTPUT	FILE DESCRIPTION
tissueCut	SN.<label>.raw.label.gef	A gene expression file in hdf5 format. It carries the expression information of the labeled tissue coverage region, with only geneExp group for the bin size of 1. The origin of expression matrix has been calibrated to (0,0), and the offset x and y has been recorded as minX and minY in the attribute of geneExp/expression dataset same to raw GEF.
	SN.<label>.label.gef	A gene expression file in hdf5 format. It includes the expression information of the labeled tissue coverage region, with geneExp group and wholeExp group in bin1, 10, 20, 50, 100, 200, and 500. It also includes a stat group. The original expression matrix has been calibrated to (0,0), and the offset x and y have been recorded as minX and minY in the attribute of geneExp/expression dataset.
	tissue_fig	This directory stores the statistical plots for the tissue coverage region
	scatter_100x100_MID_gene_counts.png	Scatter plot of MID count and gene types in each bin (bin100)
	scatter_150x150_MID_gene_counts.png	Scatter plot of MID count and gene types in each bin (bin150)
	scatter_200x200_MID_gene_counts.png	Scatter plot of MID count and gene types in each bin (bin200)
	scatter_20x20_MID_gene_counts.png	Scatter plot of MID count and gene types in each bin (bin20)
	scatter_50x50_MID_gene_counts.png	Scatter plot of MID count and gene types in each bin (bin50)
	statistic_100x100_MID_gene_DNB.png	Univariate distribution of MID count, gene types, and DNB numbers with rug along the x axis (bin100)
	statistic_150x150_MID_gene_DNB.png	Univariate distribution of MID count, gene types, and DNB numbers with rug along the x axis (bin150)
	statistic_200x200_MID_gene_DNB.png	Univariate distribution of MID count, gene types, and DNB numbers with rug along the x axis (bin200)
	statistic_20x20_MID_gene_DNB.png	Univariate distribution of MID count, gene types, and DNB numbers with rug along the x axis (bin20)
	statistic_50x50_MID_gene_DNB.png	Univariate distribution of MID count, gene types, and DNB numbers with rug along the x axis (bin50)
	violin_100x100_MID_gene.png	Violin plots show the distribution of deduplicated MID count and gene types in each bin (bin100)
	violin_150x150_MID_gene.png	Violin plots show the distribution of deduplicated MID count and gene types in each bin (bin150)
	violin_200x200_MID_gene.png	Violin plots show the distribution of deduplicated MID count and gene types in each bin (bin200)
	violin_20x20_MID_gene.png	Violin plots show the distribution of deduplicated MID count and gene types in each bin (bin20)

STEP	OUTPUT	FILE DESCRIPTION
	violin_50x50_MID_gene.png	Violin plots show the distribution of deduplicated MID count and gene types in each bin (bin50)
	tissuecut.stat	Statistical report for tissue coverage region.
	<label>.tissuecut.stat	Statistical report for labeling coverage region.
cellCut	SN.cellbin.gef	A gene expression file for cells in hdf5 format. Cellbin GEF includes the expression information of the cells, such as the coordinate of the centroid, boundary coordinates, expression of genes, and cell area. The cell is recorded by its boundary. The origin of expression matrix has been calibrated to (0,0), and the coordinate offset x and y have been recorded as offsetX and offsetY in the attribute of the GEF file, the same as minX and minY in the raw GEF file.
cellCorrect	SN.adjusted.cellbin.gef	Cellbin GEF generated from adjusted cell segmentation TIFF image and *.raw.gef.
	SN.adjusted.gem	Cellbin GEM generated from adjusted cell segmentation TIFF image and *.raw.gef.
	<stainType>_SN_mask_edm_dis_10.tif	Adjusted cell segmentation binary image in TIFF format.
spatialCluster	SN.bin200_1.0.spatial.cluster.h5ad	H5AD file for spatial clustering analysis result.
cellCluster	SN.cell.cluster.h5ad	H5AD file for cell clustering analysis result.
	SN.adjusted.cell.cluster.h5ad	H5AD file for cell clustering analysis result, based on adjusted Cellbin GEF.
saturation	plot_1x1_saturation.png	Sequencing saturation analysis plots for bin1. Calculated by 1-(unique reads/total reads) for each bin (bin1).
	plot_200x200_saturation.png	Sequencing saturation analysis plots for bin200. Calculated by 1-(unique reads/total reads) for each bin (bin1).
	sequence_saturation.tsv	Sequence saturation file. The nine colums are sampling component (#sample), bin1 total reads (bar_x), sequence saturation value at bin1 (bar_y1), median gene count at bin1 (bar_y2), unique reads at bin1 (bar_umi), bin200 total reads (bin_x), sequence saturation value at bin200 (bin_y1), median gene count at bin200 (bin_y2), and unique reads at bin200 (bin_umi), respectively.
report	SN.report.html	HTML analytical report.
	SN.statistics.json	Statistical summary report in JSON format. It gathers all important statistical metrics from the statistical report in each step.
	scatter_1x1_MID_gene_counts.png	Scatter plot of MID count and gene types in each cell.
	statistic_1x1_cell_area.png	Univariate distribution of cell area with rug along the x axis of each cell.

STEP	OUTPUT	FILE DESCRIPTION
	statistic_1x1_DNB.png	Univariate distribution of DNB numbers with rug along the x axis of each cell.
	statistic_1x1_gene.png	Univariate distribution of gene types with rug along the x axis of each cell.
	statistic_1x1_MID.png	Univariate distribution of MID count with rug along the x axis of each cell.
	violin_1x1_gene.png	Violin plots show the distribution of gene types in each cell.
	violin_1x1_MID.png	Violin plots show the distribution of deduplicated MID count in each cell.

Appendix C: Handling Errors and Exceptions

Common Errors

- **File access error:**

  Some examples of file access error:

```
terminate called after throwing an instance of 'std::invalid_argument'
what():  Could not open the file: xxxxxx
...
```

  Or

```
#006: H5FDsec2.c line 352 in H5FD__sec2_open(): unable to open file: name = '/path/to/hdf5/file', errno = 2, error message = 'No such file or directory', flags = 0, o_flags = 0
    major: File accessibility
    minor: Unable to open file
...
```

Solution:

```
# Attempt 1: Bind file path before execute command.
$ export SINGULARITY_BIND="/path/to/file/directory"

# Attempt 2: Turn off the HDF5 lock on the file by running the command below before running SAW.
$ export HDF5_USE_FILE_LOCKING=FALSE
```

mapping

- **readNameSeparator error:**

```
EXITING: FATAL INPUT ERROR: empty value for parameter “readNameSeparator” in input
“Command-Line”
SOLUTION: use non-empty value for this parameter
```

try to delete --readNameSeparator \” \” from the command.

- **limitBAMsortRAM error:**

```
Error code: SAW-A10183

EXITING because of fatal ERROR: not enough memory for BAM sorting:
SOLUTION: re-run STAR with at least -limitBAMsortRAM xxxxxxxxx
```

Modify the value specified by the --limitBAMsortRAM refer to the SOLUTION in the stderr.

- **limitOutSJcollapsed error:**

```
Error code: SAW-A10182

EXITING because of fatal error: buffer size of SJ output is too small
Solution: increase input parameter --limitOutSJcollapsed
```

the parameters --limitOutSJcollapsed and --limitIObufferSize are paired. Please replace them with --limitOutSJcollapsed 10000000 --limitIObuffer-Size=480000000 and try again.

imageTools

- **OSError raises for imageTools merge**

```
OSError: [Errno 30] Read-only file system: ‘/opt/saw_v5.4.0_software/pipeline/im-
ageTools/imagetools-1.0.0/log’
```

Error raises because the -o input is a filename without path. Try to input /path/to/output/file.rpi or ./file.rpi for -o.

Appendix D: Error Codes

SAW error codes are designed to catch error information and describe it in an easily understood way. Users can use these messages to fix the problem by themselves. The error messages will be output to the "errcode.log" file.

The error code is designed to express in three parts, date-time, error code, and explanation. The date-time information could help users differentiate errors in different program executions. The error code part provides information for the pipeline modules and error types. The explanation section displays the most explicit message stating the error/exception and possible solutions.

PIPELINE	CODE	ERROR TYPE	EXAMPLES AND ERROR HANDLING
splitMask	SAW-A00001	Parameters invalid or missing	e.g. "parameters error" Please check your input parameters. Some required parameters might be missed. e.g. "splitBcPos error, expected 1_24 or 2_25" Please check your input CID position is either 1_24 or 2_25.
	SAW-A00002	File open failed	Please check the input file exists and has the correct access permission.
	SAW-A00003	File parse failed	e.g. "only support .bin or .h5 file." Please check your input file is in the correct file format.
	SAW-A00004	Other IO API error	e.g. "cannot write to file, /path/to/file" Please check your output directory path is an existing path.
CIDCount	SAW-A00021	Parameters invalid or missing	Please check your input parameters. Some required parameters might be missed.
	SAW-A00022	File open failed	e.g. "cannot open such file, /path/to/file" Please check the input file exists and has the correct access permission.
	SAW-A00023	Failed to parser the file	Please check your input file is in the correct file format.
	SAW-A00024	Other API error	Please check Appendix C or contact FAS/FBS for help.
	SAW-A00025	Software exception	Please check Appendix C or contact FAS/FBS for help.

PIPELINE	CODE	ERROR TYPE	EXAMPLES AND ERROR HANDLING
mapping	SAW-A10101	Parameters invalid or missing	e.g. "EXITING: FATAL INPUT ERROR: unrecognized parameter name "outSAMattribute" in input "Command-Line-Initial"" Please check the spelling of your argument and parameters. e.g. "please check the umi position and length" Please check the length of the reads in FQ1 are consistent with the parameters set for barcode length and umi length. For example, the length of the sum of barcode and umi set in bcPara file is 35 bp, but one of the reads in FQ1 is 30 bp, then you will see A10101 error code. Please check the parameters set in the bcPara file are consistent with your FQs.
	SAW-A10102	File open failed	e.g. "Error, cannot open the file which be expected in gz or ascii format" Please check the file permission and file format. e.g. "barcodePositionMapFile does not exists: /path/to/mask" Please check the mask file exists and has the correct access permission.
	SAW-A10103	File parse failed	e.g. "sequence and quality have different length" Please check the completeness of the reads in FQ. This issue may arise if the FQs are in incorrect format or the file was incompletely written or transferred.
	SAW-A10104	Invalid data or data exception	Error data. Please check the file format and content.
	SAW-A10105	File deletion failed	This error arose if "_STARtmp" directory failed to be deleted. Please check whether the program has completely finished according to the *.Log.progress.out or *.run.log file.
	SAW-A10106	File IO failed	Please contact FAS/FBS for help.
	SAW-A10107	Failure on APIs of system and libraries	Please contact FAS/FBS for help.
	SAW-A10108	Software assert	Please contact FAS/FBS for help.
	SAW-A10109	Software exception	Please contact FAS/FBS for help.
	SAW-A10110	Allocate memory error	This error occurs when you run out of RAM. You may need to kill some jobs that use the same RAM or upgrade your hardware to get more RAM resources for your job.

PIPELINE	CODE	ERROR TYPE	EXAMPLES AND ERROR HANDLING
mapping	SAW-A10111	Out of disk space	This error occurs when you store too many files on your hard disk. Please remove some files to free disk space.
	SAW-A10200	Parameter missing	Please check your input parameters. Some required parameters might be missed.
	SAW-A10201	CID comparison rate is too low	This error usually arises because the CID information of the input FQs is not the same as the CID in the input mask file. Please use the correct SN-FQ pairs for `mapping`.
	SAW-A10202	Fail to create the index for the BAM file	Please contact FAS/FBS for help.
	SAW-A10300	Fail to load indexed reference	Please check the existence, access permission, and completeness for the indexed reference.
	SAW-A10301	STAR parameter missing	Please check whether basic STAR parameters have been set.
count	SAW-A20001	Parameter missing	Please check your input parameters. Some required parameters might be missed.
	SAW-A20002	File open failed	Please check the input file exists and has the correct access permission.
	SAW-A20003	File parse failed	Please check your input BAM header is in the correct file format.
	SAW-A20004	Allocate memory error	This error occurs when you run out of RAM. You may need to kill some jobs that use the same RAM or upgrade your hardware to get more RAM resources for your job.
	SAW-A20005	File parse failed	e.g. "Found <number> gene names with their length exceeding 64 characters" Please check gene names in your input GTF/GFF file.
	SAW-A20101	Parameter missing	Please check your input parameters. Some required parameters might be missed.
	SAW-A20102	File open failed	Please check the file that exists and has the correct access permission.
	SAW-A20103	File parse failed	Please check your input file has a valid column number.
	SAW-A20105	File parse failed	e.g. "Found <number> gene names with their length exceeding 64 characters" Please check gene names in your input GTF/GFF file.

PIPELINE	CODE	ERROR TYPE	EXAMPLES AND ERROR HANDLING
merge	SAW-A30001	Parameter missing	Please check your input parameters. Some required parameters might be missed.
	SAW-A30002	File open failed	Please check the input file exists and has the correct access permission.
	SAW-A30003	File parse failed	Please check your mask file is in the correct file format. Only support .h5/.bin mask file.
	SAW-A30004	Allocate memory error	Please check whether the range of the coordinates is too large, or you have run out of RAM. You may need to kill some jobs that use the same RAM or upgrade your hardware to get more RAM resources for your job.
	SAW-A30005	Fail to open input file	Fail to open input TXT file. Please check the file that exists and have the correct access permission.
register	SAW-A40001	Parameter missing	Please check your input parameters. Some required parameters might be missed.
	SAW-A40002	File open failed	Please check the input file that exists and has the correct access permission. Or, please check whether a stitched panoramic TIFF (.tif or .tiff) image exists in the TAR.GZ.
	SAW-A40003	File parse failed	Please check your input file is in the correct file format. The -v input gene expression matrix should be either a *tsv, barcode_gene_exp.txt, *.gem.gz, or *raw.gef.
	SAW-A40004	Invalid data or data exception	Error data. Please check the file content. This error may arise because the -v input file is empty, the CZI file in TAR.GZ is invalid, or the QCPassFlag in IPR is 0.
	SAW-A40005	Tissue segmentation error	Abnormal tissue segmentation reference score in image preprocessing.
	SAW-A40006	IPR field missing	"Stitch/BGIStitch/StitchedGlobalLoc", does not exist.
	SAW-A40007	Tiled image missing	Please check the uncompressed image folder has tiled images.
	SAW-A40008	Insufficient GPU memory	GPU resources in the current node are insufficient. Please redeliver the task on an adequate one.

PIPELINE	CODE	ERROR TYPE	EXAMPLES AND ERROR HANDLING
`register`	SAW-A40009	Abnormal chip SN prefix	Check "ImageInfo -> STOmicsChipSN" in.ipr where it offers the information of chip SN prefix.
`imageTools`	SAW-A40401	Parameter missing	Please check your `imageTools merge` input parameters. Some required parameters might be missed.
	SAW-A40402	File open failed	Please check the `imageTools merge` input file that exists and has the correct access permission.
	SAW-A40405	Invalid input	`imageTools merge` inputs of less than two images or more than three images.
	SAW-A40406	File pairing failed	Please check the `imageTools merge` input TIFF sizes are the same. Since the `merge` function is used for evaluating segmentation results, the input images are supposed to be the same in size and position (tissue position in the whole image).
	SAW-A40501	Parameter missing	Please check your `imageTools overlay` input parameters. Some required parameters might be missed.
	SAW-A40502	File open failed	Please check the `imageTools overlay` input file that exists and has the correct access permission.
	SAW-A40504	Invalid data or data exception	Error data. Please check the file content. This error may arise because the -c input IPR file does not include Stitch/TransformTemplate or Register/MatrixTemplate information.
	SAW-A40601	Parameter missing	Please check your `imageTools img2rpi` input parameters. Some required parameters might be missed.
	SAW-A40602	File open failed	Please check the `imageTools img2rpi` input file that exists and has the correct access permission.
	SAW-A40605	Invalid input	`imageTools img2rpi` input -i and -g have different length. These two inputs are supposed to be paired.
	SAW-A40701	Parameter missing	Please check your `imageTools ipr2img` input parameters. Some required parameters might be missed.

PIPELINE	CODE	ERROR TYPE	EXAMPLES AND ERROR HANDLING
imageTools	SAW-A40702	File open failed	Please check the `imageTools ipr2img` input file exists and has the correct access permission. Or, please check whether a stitched panoramic TIFF (.tif or .tiff) image exists in the TAR.GZ.
	SAW-A40703	File parse failed	Please check your `imageTools ipr2img` input file is in the **ImageQC/ImageStudio** output TAR.GZ format.
	SAW-A40704	Invalid data or data exception	Error data. Please check the `imageTools ipr2img` input IPR file content. This error may arise because the CZI file in TAR.GZ is invalid, or the image has not either automatically or manually registered with the expression matrix. The second circumstance can be confirmed from IPR by checking whether the StereoResepSwitch/register is TRUE (has not performed automatic registration) or the ManualState/register is FALSE (has not performed manual registration). The third possible reason is the StereoResepSwitch/tissueseg is TRUE, therefore cannot output cell segmentation result.
	SAW-A40706	File pairing failed	Please check the shape of registered tissue segmentation images stored in the `imageTools ipr2img` input IPR are the same. Or please contact FAS/FBS for help.
tissueCut	SAW-A50001	Parameter missing	Please check your input parameters. Some required parameters might be missed.
	SAW-A50002	File open failed	Please check the file that exists and has the correct access permission.
	SAW-A50003	File parse failed	Please check your input file is in the correct file format.
	SAW-A50004	Fail to create output file	Fail to create output file. Please check your writing permission of the output directory.
	SAW-A50005	Fail to write TIFF	Fail to write a TIFF file.
	SAW-A50006	h5AttrWrite error	Please check the H5 file attributes.
	SAW-A50007	h5DatasetWrite error	Please check the H5 file dataset.
	SAW-A50008	Fail to create TIFF	Please check the access permission to write a TIFF image.
	SAW-A50009	Different sizes between TIFF and GEF	Please check the sizes of the TIFF image and GEF (gene expression matrix) respectively. Make sure they are the same size.
	SAW-A50010	Allocate memory error	This error occurs when you run out of RAM. You may need to kill some jobs that use the same RAM or upgrade your hardware to get more RAM resources for your job.

PIPELINE	CODE	ERROR TYPE	EXAMPLES AND ERROR HANDLING
cellCut	SAW-A60001	Parameter missing	Please check your input parameters. Some required parameters might be missed.
	SAW-A60002	File open failed	Please check the file that exists and has the correct access permission.
	SAW-A60003	File parse failed	The file does not contain correct information. Please check the file format.
	SAW-A60110	Program version error	Please check your output GEF version. Your input GEF version might be too old.
	SAW-A60111	Call process error	e.g. "Please call freeRestriction first, or call restrictRegion function before restrictGene." This error arose because the invocation flow order was messed up. Please modify your invocation flow as prompted.
	SAW-A60120	Invalid data or data exception	Error data. Please check the file content.
	SAW-A60121	File information missing	Failed to read the file. Please check whether the file is damaged.
	SAW-A60122	File pairing failed	Please check the TIFF mask size is consistent with the size of expression matrix. Since the mask has been registered with the expression matrix, their sizes are supposed to be the same.
	SAW-A60130	Fail to create output file	Fail to create output H5 file. Please check your writing permission of the output directory or contact FAS/FBS for help.
	SAW-A60140	Allocate memory error	This error occurs when you run out of RAM. You may need to kill some jobs that use the same RAM or upgrade your hardware to get more RAM resources for your job.
	SAW-A60150	Dimensions of gene expression matrix did not match	Please contact FAS/FBS for help.

PIPELINE	CODE	ERROR TYPE	EXAMPLES AND ERROR HANDLING
spatial-Cluster	SAW-A70001	Parameter missing	e.g. "-i or --gef_file is missing" Please check your input parameters. Some required parameters might be missed.
	SAW-A70002	File open failed	e.g. “cannot access /path/to/file: No such file or directory.” Please check the file that exists and has the correct access permission.
	SAW-A70005	Value error	e.g. "The bin size is out of range, please check the range of gef binsize is in [1,10,20,50,100,200,500]." Please reset the bin size as prompted. e.g. "Gene number less than 3000, please check your gef file" Please check the content of your GEF file, and make sure there are at least 3000 genes for clustering.
cellClus-ter	SAW-A70101	Parameter missing	e.g. "-i or --gef_file is missing" Please check your input parameters. Some required parameters might be missed.
	SAW-A70102	File open failed	e.g. “cannot access /path/to/file: No such file or directory.” Please check the file that exists and has the correct access permission.
	SAW-A70105	Value error	e.g. "The bin size is out of range, please check the range of gef binsize is in [1,10,20,50,100,200,500]." Please reset the bin size as prompted. e.g. "Gene number less than 3000, please check your gef file" Please check the content of your GEF file, and make sure there are at least 3000 genes for clustering.

PIPELINE	CODE	ERROR TYPE	EXAMPLES AND ERROR HANDLING
saturation	SAW-A80001	Parameter missing	e.g. "-i is missing." Please check your input parameters. Some required parameters might be missed.
	SAW-A80002	File open failed	Please check the file that exists and has the correct access permission.
	SAW-A80003	File parse failed	Invalid GEF file. Please check the file format.
	SAW-A80004	Invalid data or data exception	e.g. "no data left after filter by coordinates." Please check the file content.
	SAW-A80005	Invalid data or data exception	e.g. "total map reads is 0, please check file format from --bcstat" Please check the file content of the input file as prompted.
	SAW-A80006	File pairing failed	e.g. "map reads less than annotated reads." Please check the input `mapping` statistical report and the `count` statistical report are from the same analysis.
	SAW-A80007	Plot error	Please contact FAS/FBS for help. PATH environment may not have python3.
	SAW-A80008	Fail to create output	Fail to generate saturation file, please contact FAS/FBS for help.

PIPELINE	CODE	ERROR TYPE	EXAMPLES AND ERROR HANDLING
report	SAW-A90001	Parameter missing	e.g. "-m or --barcodeMapStat is missing." Please check your input parameter as prompted. Some required parameters might be missed.
	SAW-A90002	File open failed	e.g. "cannot access *: No such file or directory." Please check the file that exists and has the correct access permission.
	SAW-A90003	File parse failed	JSON file format error. This error may arise because the input statistics files were not generated in the same SAW version as `report`. Or, the input `mapping` file prefix can not be parsed. Please contact FAS/FBS for help.
	SAW-A90004	Invalid data or data exception	e.g."information loss: fail to find 'bin_[size]' or 'ssDNA' in '*.rpi'." Please check the file content.
	SAW-A90005	Fail to create output file	Fail to create output file. Please check your writing permission of the output directory.
manualRegister	SAW-A40801	Parameter missing	Please check your input parameters. Some required parameters might be missed.
	SAW-A40802	File open failed	Please check the file that exists and has the correct access permission. Or, please check whether the pre-registered image fov_stitched_transformed.tif exists in the input directory.
	SAW-A40803	File parse failed	Please check your input file is in the correct file format. The -v input gene expression matrix should be either a *tsv, barcode_gene_exp.txt, *.gem.gz, or *raw.gef.
	SAW-A40804	Invalid data or data exception	Error data. Please check the file content. This error may arise because the -v input file is empty. The second possible reason is that the gene expression matrix information in the IPR Register module (MatrixShape, Xstart, Ystart) does not match with the input GEF file (minX, minY, maxX, maxY), because the manual registration has to be processed on the identical matrix.

PIPELINE	CODE	ERROR TYPE	EXAMPLES AND ERROR HANDLING
cellCorrect	SAW-A13001	GEF parse failed	e.g. "-i parameter is invalid, please check your input." Please check the path or file format of GEF.
	SAW-A13002	TIFF parse failed	e.g. "-m parameter is invalid, please check your input." Please check the path or size of TIFF image.
	SAW-A13003	Fail to create output	e.g. "-o parameter is invalid, please check your input." Please check the access permission to write files or whether the output directory exists.
	SAW-A13004	Invalid adjusting distance	e.g. "-d parameter exceeds the range of adjusting distance." Please check whether the adjusting distance is in a proper and reasonable range.
lasso	SAW-A00031	Parameters invalid or missing	e.g. "-i/-o/-m/-n is missing" Please check your input parameters. Some required parameters might be missed.
	SAW-A00032	File open failed	e.g. "cannot access *: No such file or directory." Please check the file that exists and has the correct access permission.
	SAW-A00033	File parse failed	Ee.g. "file type error: * is not GEF/GEOJSON file." Invalid file. Please check the file format.

References

1. STOmics/SAW. Accessed April 17, 2023. https://github.com/STOmics/SAW
2. Chen A, Liao S, Cheng M, et al. Large field of view-spatially resolved transcriptomics at nanoscale resolution Short title: DNA nanoball stereo-sequencing. *bioRxiv.* Published online January 24, 2021:2021.01.17.427004. doi:10.1101/2021.01.17.427004
3. Cock PJA, Fields CJ, Goto N, Heuer ML, Rice PM. The Sanger FASTQ file format for sequences with quality scores, and the Solexa/Illumina FASTQ variants. *Nucleic Acids Res.* 2009;38(6):1767-1771. doi:10.1093/nar/gkp1137
4. Archives SR, Sra T, Nucleotide I, et al. File Format Guide 1. Published online 2009:1-11. Accessed May 21, 2021. https://www.ncbi.nlm.nih.gov/sra/docs/submitformats/
5. Merkel D. Docker: lightweight Linux containers for consistent development and deployment. *Linux J.* 2014;2014(239):2. Accessed October 15, 2021. https://www.linuxjournal.com/content/docker-lightweight-linux-containers-consistent-development-and-deployment
6. Kurtzer GM, Sochat V, Bauer MW. Singularity: Scientific containers for mobility of compute. *PLoS One.* 2017;12(5):e0177459. doi:10.1371/journal.pone.0177459
7. *Sequence Alignment/Map Format Specification.* Accessed May 21, 2021. https://github.com/samtools/hts-specs.
8. Dobin A, Davis CA, Schlesinger F, et al. STAR: Ultrafast universal RNA-seq aligner. *Bioinformatics.* 2013;29(1):15-21. doi:10.1093/bioinformatics/bts635
9. Ensembl. GFF/GTF File Format. Published 2020. Accessed May 27, 2021. http://www.ensembl.org/info/website/upload/gff.html?redirect=no
10. GFF2 - GMOD. Accessed May 27, 2021. http://gmod.org/wiki/GFF2
11. STOmics/Stereopy: Spatial Transcriptomics Analysis in Python. Accessed May 15, 2023. https://github.com/STOmics/Stereopy
12. STOmics/geftools: Tools for manipulating GEFs. Accessed May 30, 2023. https://github.com/STOmics/geftools
13. STOmics/gefpy: gef io, draw out from Stereopy. Accessed May 30, 2023. https://github.com/STOmics/gefpy

Contact Us

Beijing Genomics Institute (Shenzhen)

https://en.stomics.tech

Email: info_global@stomics.tech

Please raise GitHub issues for reporting bugs and requesting features:

SAW GitHub Issue Page: https://github.com/STOmics/SAW/issues