STOmics

STEREO-SEQ ANALYSIS WORKFLOW SOFTWARE SUITE

USER MANUAL

Software Version: V5.4.0

Manual Version: A4

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 micrographs; 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;

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

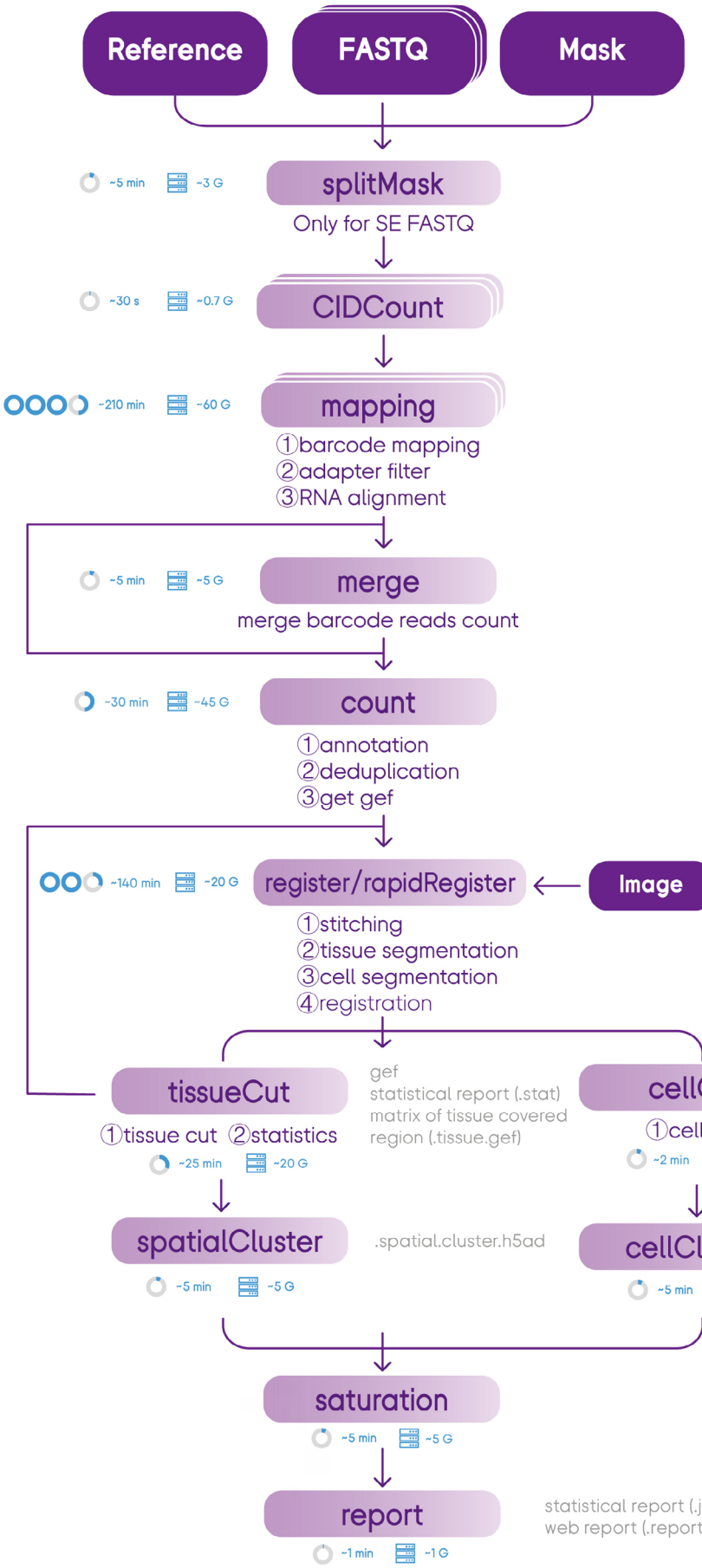
WORKFLOW

1 G reads

64 bit CentOS/RedHat 7.8
64 bit Ubuntu 20.04

Minimum requirements: 8 cores, 128 G, 1 TB

Higher requirements: 12 cores, 256 G, 1 TB



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)

cellbin.gef

statistical report (.json)
web report (.report.html)

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 SE FASTQ files.
- **CIDCount:** Counting CIDs in the Stereo-seq Chip T mask file and roughly estimating memory required to do `mapping`. (Highly suggested to run this before `mapping`.)
- **mapping:** Corresponds *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):** Combines CID (same as barcodes) listed files with reads count from multiple runs of `mapping`. Only applicable for an analysis that requires to combine multiple pairs of FASTQ.
- **count:** Reads 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 failed QC or input ssDNA 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 failed QC or input ssDNA 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 covered area generated from `tissueCut`.
- **cellCut:** Identify cell coverage area on the ssDNA plot and extract gene expression matrix of the corresponding spatial location by taking inputs from both `count` and `register` pipeline. Optional module when image failed QC or input ssDNA image is absent.
- **cellCluster:** Perform clustering analysis for cell bins according to the gene expression matrix which is generated from `cellCut`. Optional module when image failed QC or input ssDNA 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 integrate the analysis result from each step, as well as an HTML web analysis report, showing spatial expression distribution of genes, key statistical metrices, 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:** Manipulating GEF file.
- **rapidRegister:** Run registration without performing cell segmentation.
- **checkGTF:** Check GTF/GFF file is prepared in the correct format, otherwise, re-format one that meet the compatibility requirements for `count`.
- **Other applications of image Tools:** 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 the evaluation of stitching and registration result. 3) Write TIFF images into RPI format.

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

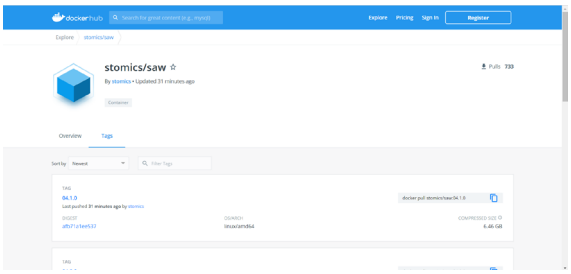
SAW >= 5.0.0 requires imageQC version >= 1.1.0 that provide IPR file for recording image processing data.

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 5.4.0 as an example.

1.4.1. SAW Installation via Singularity

CRITICAL STEPS: Please replace the red highlighted inputs with your own data path.

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

```
$ singularity build SAW_v5.4.0.sif docker://stomics/saw:05.4.0  ## option 1
$ singularity build --sandbox SAW_v5.4.0/ docker://stomics/saw:05.4.0  ## 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_v5.4.0/ docker://stomics/saw:05.4.0
```

Step 2: Run pipelines (3 options):

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_v5.4.0.sif <application>  ## option 1.1
$ singularity exec /path/to/SAW_v5.4.0.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_v5.4.0.sif /bin/bash  ## option 2.1
Singularity>
Singularity> <shell-command>
Singularity> exit
$
$ singularity shell /path/to/SAW_v5.4.0.sif  ## option 2.2
Singularity>
Singularity> <shell-command>
Singularity> exit
$
$ singularity shell SAW_v5.4.0  ## 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_v5.4.0.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:05.4.0
```

Step 2: Run pipelines

```
$ docker run -d -v /path/to/data:stomics/saw:05.4.0 /bin/sh -c "<shell-command>"
```

1.5. SAW Github

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

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.

Run **splitMask** requires the following files:

- Stereo-seq Chip T Mask file **(.h5 or .bin)**

Expected running time for ~1G reads: ~5 min, Memory: ~3G

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 or .bin).
[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 SE 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 start from the 2nd base to the 25th base that are used for splitting mask file.

2.1.2 Usage Example

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_v5.4.0.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:

```
$ 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**.

Run **CIDCount** requires the following files:

- Stereo-seq Chip T Mask file **(.h5 or .bin)**

Expected running time for ~1G reads: ~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	(Required) A string of species name.
-g	(Required) Genome file size in GB.

2.2.2 Usage Example

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

```
$ singularity exec SAW_v5.4.0.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 SE 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.

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

```
$ singularity exec SAW_v5.4.0.sif CIDCount \
      -i /path/to/output/00.splitmask/{index}.{SN}.barcodeToPos.bin \  ## 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}”
```

2.2.3 Outputs

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

```
$ singularity exec SAW_v5.4.0.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 will be added based on the records of the Mask file. 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 with 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.

Run **mapping** requires the following files:

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

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

NOTE! Before proceeding, if users need to estimate the number of CIDs and the memory needed for running mapping, please refer to 2.2 CIDcount for more information.

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.

Parameter	Function
`--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) Name of a parameters file 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.
`--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.
`--outBAMsortingBinsN`	(STAR option; defaults to 50) number of genome bins for coordinate-sorting. If the read2 FASTQ file size is greater than 200, it is better to set this value to 100.

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 PE sequences specify the path to the FASTQ file of read1 here. For SE sequences, input FASTQ file with the same split index as the mask file. A more elegant way to input a long list of SE FASTQs is to organize them in a **FQ_{index}.list** file. Please check **2.3.2 Usage Example SE scenario 2** to get more information.
`in2`	(Optional) Path to the FASTQ file of read2. Only valid for PE sequencing.
`encodeRule`	(Required) Encoding rule for the four bases. ACTG stands for A->0, C->1, T->2, G->3.
`out`	(Optional) Set output file prefix.
`action`	(Required; defaults to 1) Action number. Set to 4 in `mapping` Valid options: 1=CID stat, 2=CID overlap, 3=get CID position map, 4=map CID to slide, or 5=merge CID list.
`barcodeReadsCount`	(Required) Mapped CID list file with reads counts for each CID. This is an output file in `mapping`
`platform`	(Optional) Sequencing platform. Valid options include but not limited to: **SEQ500, G400, T1, T7, T10**.
`barcodeStart`	(Required; defaults to 0) CID start position. Set to 0 for PE, 1 for SE.
`barcodeLen`	(Required; defaults to 25) CID length. Set to 25 for PE, 24 for SE.
`umiStart`	(Required; defaults to 25) MID start position.
`umiLen`	(Required; defaults to 10) MID length.
`umiRead`	(Required; defaults to 1) Declare the read contains MID.
`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=15`	(Optional) Number of consecutive A in the read that will be trimmed. Recommend to set this value to 15.
`mismatchInPolyA=2`	(Optional) Number of mismatch bases in searching poly A. Recommend to set this value to 2

2.3.2 Usage Example

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

- **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.**

PE scenario 1: Prepare **{lane}.bcPara** file for PE 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
encodeRule=ACTG
out={lane}
barcodeReadsCount=/path/to/ouptut/01.mapping/{lane}.barcodeReadsCount.txt
action=4
platform=T10
barcodeStart=0
barcodeLen=25
umiStart=25
umiLen=10
umiRead=1
mismatch=1
bcNum=645784920  ## Input the first line from output of CIDCount
polyAnum=15
mismatchInPolyA=2
```

PE scenario 2: Prepare **{lane}.bcPara** file for PE 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
encodeRule=ACTG
out={lane}
barcodeReadsCount=/path/to/multi_lane_output/01.mapping/{lane}.barcodeReadsCount.txt
action=4
platform=T10
barcodeStart=0
barcodeLen=25
umiStart=25
umiLen=10
umiRead=1
mismatch=1
bcNum=645784920  ## Input the first line from output of CIDCount
polyAnum=15
mismatchInPolyA=2
```

Run **mapping** for PE input:

```
$ singularity exec SAW_v5.4.0.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}_barcodeMap.stat
```

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

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

Note! **{index}** need to be replaced by the index of the split mask file which is corresponding to the index of the SE 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}
out={SN}.bc.out${index}
barcodeReadsCount=/path/to/ouptut/01.mapping/{index}.barcodeReadsCount.txt
action=4
platform=T10
barcodeStart=1
barcodeLen=24
umiStart=25
umiLen=10
umiRead=1
mismatch=1
bcNum=38284877  ## Input the first line from output of CIDCount
polyAnum=15
mismatchInPolyA=2
```

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

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, if a library contains 2 barcodes that sequenced in the same lane, the **{index}** and **{idx}** for the third FASTQ file of barcode 1 are 01 and 03, respectively.

```
$ 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}
out={SN}.bc.out${idx}
barcodeReadsCount=/path/to/ouptut/01.mapping/{idx}.barcodeReadsCount.txt
```

```
action=4
platform=T10
barcodeStart=1
barcodeLen=24
umiStart=25
umiLen=10
umiRead=1
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.


```
$ cat SN_SE_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/{idx}.bcPara
in=/path/to/output/00.splitmask/{index}.{SN}.barcodeToPos.bin  ## split mask file
in1=/path/to/output/{SN}_SE_fastq_{index}.list
out={SN}.bc.out${idx}
barcodeReadsCount=/path/to/output/01.mapping/{idx}.barcodeReadsCount.txt
action=4
platform=T10
barcodeStart=1
barcodeLen=24
umiStart=25
umiLen=10
umiRead=1
mismatch=1
bcNum=38284877 ## Input the first line from output of CIDCount
polyAnum=15
mismatchInPolyA=2
```

Run **mapping** pipeline

```
$ singularity exec SAW_v5.4.0.sif mapping \
                --outSAMattributes spatial \
                --outSAMtype BAM SortedByCoordinate \
```

```
                --genomeDir /path/to/genomeDir \
                --runThreadN 8 \
                --outFileNamePrefix /path/to/output/01.mapping/{index}. \  ## {in-
dex} 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}_barcodeMap.stat  ## {index} or
{idx} depending on the scenario
```

2.3.3 Outputs

PE 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.Aligned.sortedByCoord.out.bam.csi
├── lane_barcodeMap.stat
├── lane.barcodeReadsCount.txt
├── lane.bcPara
├── lane.Log.final.out
├── lane.Log.out
├── lane.Log.progress.out
└── lane.SJ.out.tab
```

PE scenario 2 output files are organized as below (Here showing 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
└── 01.mapping
    ├── lane1.Aligned.sortedByCoord.out.bam
    ├── lane1.Aligned.sortedByCoord.out.bam.csi
    ├── lane1_barcodeMap.stat
    ├── lane1.barcodeReadsCount.txt
    ├── lane1.bcPara
    ├── lane1.Log.final.out
    ├── lane1.Log.out
    ├── lane1.Log.progress.out
    ├── lane1.SJ.out.tab
    ├── lane2.Aligned.sortedByCoord.out.bam
    ├── lane2.Aligned.sortedByCoord.out.bam.csi
    ├── lane2_barcodeMap.stat
    ├── lane2.barcodeReadsCount.txt
    ├── lane2.bcPara
```

```
    ├── lane2.Log.final.out
    ├── lane2.Log.out
    ├── lane2.Log.progress.out
    └── lane2.SJ.out.tab
```

SE 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.mapping
    ├── 01.Aligned.sortedByCoord.out.bam
    ├── 01.Aligned.sortedByCoord.out.bam.csi
    ├── 01_barcodeMap.stat
    ├── 01.barcodeReadsCount.txt
    ├── 01.bcPara
    ├── 01.Log.final.out
    ├── 01.Log.out
    ├── 01.Log.progress.out
    ├── 01.SJ.out.tab
    ...
    ├── 16.Aligned.sortedByCoord.out.bam
    ├── 16.Aligned.sortedByCoord.out.bam.csi
    ├── 16_barcodeMap.stat
    ├── 16.barcodeReadsCount.txt
    ├── 16.bcPara
    ├── 16.Log.final.out
    ├── 16.Log.out
    ├── 16.Log.progress.out
    └── 16.SJ.out.tab
```

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

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

2.4 merge (optional)

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

Run **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 reads 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_v5.4.0.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 SE 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}.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 statistics information for the overall summarization result. Through quantification of annotated reads, **count** generates spatial gene expression data after de-duplicate 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 to differentiate genomic features, **count** also output exon MID count separately.

Run **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.
--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**).
-m	(Optional; defaults to detected) Set available memory (GB).
--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_v5.4.0.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 8 \
        -m 128
```

* 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/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_v5.4.0.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 8 \
        -m 128
```

* 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_v5.4.0.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 8 \
        -m 128 \
        --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
```

2.6 register

SAW **register** pipeline aligns the microscopic tissue staining image with the plot of the 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. 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 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. Image stitching and segmentation modules can be run separately with registration.

Run **register** requires the following files:

- **count** output gene expression matrix file **(.raw.gef)**
- ImageQC processed microscopic tissue staining image file **(.tar.gz)**
- ImageQC Image process record file **(.ipr, require imageQC version >= 1.1.0)**

Expected running time for 1 cm x 1 cm Stereo-seq Chip T image: ~2 h, Memory: ~20 G

Cell segmentation is the most time consuming part. You may run `rapidRegister` to skip cell segmentation, but still perform stitching, tissue segmentation and registration. Please check 2.14.2 rapidRegister to get more information about `rapidRegister`.

2.6.1 Arguments and Options

Table 2-7 `register` Arguments and Options

Parameter	Function
-i	(Required) ImageQC processed staining image TAR.GZ file or image pre-processed output directory.
-c	(Required) ImageQC 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.

2.6.2 Usage Example

Scenario 1: Process images from ImageQC output raw 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_v5.4.0.sif register \
                    -i ${image4register} \
                    -c ${imageQC} \
                    -v /path/to/output/03.count/{SN}.raw.gef \
                    -o /path/to/output/04.register
```

Scenario 2: Process images from ImageQC output raw 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_v5.4.0.sif register \
    -i ${image4register} \
    -c ${imageQC} \
    -o /path/to/output/04.register
```

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_v5.4.0.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
```

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
├── attrs.json
├── fov_stitched_transformed.tif
├── SN_date_time_version.ipr
├── SN_tissue_bbox.csv
└── transform_thumb.png
```

`register` output files of scenario 2:

```
$ tree /path/to/output/04.register
/path/to/output/04.register
...  ## skip setting files, logs and image folder
├── attrs.json
├── fov_stitched_transformed.tif
├── SN_date_time_version.ipr
└── transform_thumb.png
```

`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_date_time_version.ipr
└── SN_tissue_bbox.csv
```

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`** output 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.

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

- ImageQC processed microscopic tissue staining image file **(.tar.gz)**
- **`register`** processed Image process record file **(.ipr)**

Expected running time for ~1G reads: ~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 processed staining image TAR.GZ file.
	`-c`	(Required) ImageQC 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_v5.4.0.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 `-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
├── attrs.json
├── fov_stitched_transformed.tif
├── matrix_template.txt
├── SN_date_time_version.ipr
├── SN_mask.tif
├── SN_regist.tif
├── SN.rpi
├── SN_tissue_bbox.csv
├── SN_tissue_cut.tif
├── transform_template.txt
└── transform_thumb.png
```

imageTools ipr2img output files (if **-i** and **-o** are two different paths) are organized as below:

```
$ tree /path/to/output/04.register
/path/to/output/04.register
...  ## skip logs and image folder
├── fov_stitched_transformed.tif
├── matrix_template.txt
├── SN_mask.tif
├── SN_regist.tif
├── SN.rpi
├── SN_tissue_bbox.csv
├── SN_tissue_cut.tif
└── transform_template.txt
```

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. Users may generate registered image in TIFF or JPG format from image pyramid RPI file using open sourced python toolkit -- Stereopy [11].

If the output of tissueCut doesn't match the morphology of the tissue, user could use Stereopy to do lasso selection interactively to extract the expression matrix of tissue-covered region. Please check the tutorial, Stereopy->Examples->Interactive(https://stereopy.readthedocs.io/en/latest/Tutorials/interactive_cluster.html).

Run **tissueCut** requires the following files:

- Mapped CID list file **(.txt)**
- **count** output gene expression matrix file **(.raw.gef)**
- Directory stores aligned microscopic staining image (optional)

Expected running time for ~1G reads: ~25 min, Memory: ~20 G

2.8.1 Arguments and Options

Table 2-9 tissueCut Arguments and Options

Parameter	Function
--dnbfile	(Required) Mapped CID list file with reads counts for each CID. Input merged mapped CID list file if more than one list file were 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 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**.

2.8.2 Usage Example

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

```
$ tissueMaskFile=$(find /path/to/output/04.register -maxdepth 1 -name {SN}_
tissue_cut.tif | head -1)
$ mkdir -p /path/to/output/05.tissuecut
$ singularity exec SAW_v5.4.0.sif tissueCut \
    --dnbfile /path/to/output/02.merge/{SN}.barcodeReadsCount.txt \
    -i /path/to/output/03.count/{SN}.raw.gef \
    -o /path/to/output/05.tissuecut \
    -s ${tissueMaskFile} \
    --sn {SN} -O Transcriptomics -d
```

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

```
$ mkdir -p /path/to/output/05.tissuecut
$ singularity exec SAW_v5.4.0.sif tissueCut \
    --dnbfile /path/to/output/02.merge/{SN}.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:

```
$ 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_50x50_MID_gene_counts.png
    ├── statistic_100x100_MID_gene_DNB.png
    ├── statistic_150x150_MID_gene_DNB.png
    ├── statistic_200x200_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_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
├── bin1_mask.tif  ## tissue mask file
├── 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_50x50_MID_gene_counts.png
    ├── statistic_100x100_MID_gene_DNB.png
    ├── statistic_150x150_MID_gene_DNB.png
    ├── statistic_200x200_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_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 on highly variable genes, 3) compute the neighborhood graph and embed neighborhood graph using UMAP, 4) and clustering by Leiden algorithm.

Run **`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.

2.9.2 Usage Example

```
$ mkdir -p /path/to/output/06.spatialcluster
$ singularity exec SAW_v5.4.0.sif spatialCluster \
    -i /path/to/output/05.tissuecut/{SN}.tissue.gef \
    -o /path/to/output/06.spatialcluster/{SN}.spatial.cluster.h5ad \
    -s 200
```

2.9.3 Outputs

spatialCluster output files are:

```
$ tree /path/to/output/06.spatialcluster
/path/to/output/06.spatialcluster
└── SN.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** 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.

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
$ singularity exec SAW_v5.4.0.sif cellCut cgef \
        -i /path/to/output/03.count/{SN}.raw.gef \
        -m /path/to/output/04.register/{SN}_mask.tif \
        -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 cellCluster

SAW **cellCluster** pipeline runs by clustering cells using the Leiden algorithm similarly to the procedures of **spatialCluster**.

Run **cellCluster** requires the following files.

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

Expected running time for ~1G reads: ~5 min, Memory: ~25 G

2.11.1 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 for the clustering result in H5AD format.

2.11.2 Usage Example

```
$ mkdir -p /path/to/output/061.cellcluster
$ singularity exec SAW_v5.4.0.sif cellCluster \
        -i /path/to/output/051.cellcut/{SN}.cellbin.gef \
        -o /path/to/output/061.cellcluster/{SN}.cell.cluster.h5ad
```

2.11.3 Outputs

cellCluster output files:

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

2.12 saturation

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

- Run **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

2.12.1 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.

2.12.2 Usage Example

```
$ mkdir -p /path/to/output/07.saturation
$ singularity exec SAW_v5.4.0.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}_barcodeMap.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_v5.4.0.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}_barcodeMap.stat,/path/
to/multi_lane_output/01.mapping/{lane2}_barcodeMap.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.

2.12.3 Outputs

saturation output files:

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

2.13 report

SAW **report** pipeline is performed to integrate analysis report 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 result, and image processing information.

Run **report** requires the following files and information:

- **mapping** output statistical reports 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

2.13.1 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.

Parameter	Function
--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.
--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.
--bin1Saturation	(Optional) `saturation` output bin 1 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.

2.13.2 Usage Example

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

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_v5.4.0.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_v5.4.0.sif report \
      -m /path/to/output/01.mapping/{lane}_barcodeMap.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}.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
\
      --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 \
      -r standard_version \
      -i /path/to/output/04.register/{SN}.rpi \
      -s {SN} \
      --pipelineVersion SAW_v5.4.0 \
      --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_v5.4.0.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_v5.4.0.sif report \
      -m /path/to/multi_lane_output/01.mapping/{lane1}_barcodeMap.stat,/path/to/
multi_lane_output/01.mapping/{lane2}_barcodeMap.statt \
      -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}.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 \
      --bin50Saturation /path/to/multi_lane_output/05.tissuecut/tissue_
fig/scatter_50x50_MID_gene_counts.png \
      --bin50violin /path/to/multi_lane_output/05.tissuecut/tissue_fig/violin_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 \
      -r standard_version \
      -i /path/to/multi_lane_output/04.register/{SN}.rpi \
      -s {SN} \
      --pipelineVersion SAW_v5.4.0 \
```

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

```
      --iprFile ${imageIPR} \
      --species {species_name} \
      --tissue {tissue_type} \
      --reference {reference_index} \
      -o /path/to/multi_lane_output/08.report
```

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

```
$ mkdir -p /path/to/output/08.report
$ singularity exec SAW_v5.4.0.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_v5.4.0.sif report \
      -m /path/to/output/01.mapping/{lane}_barcodeMap.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}.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
\
      --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 \
```

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

```
    -r standard_version \
    -s {SN} \
    --pipelineVersion SAW_v5.4.0 \
    --species {species_name} \
    --tissue {tissue_type} \
    --reference {reference_index} \
    -o /path/to/output/08.report
```

2.13.3 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_MID_gene_DNB.png
└── violin_1x1_MID_gene.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
```

2.14 Other Applications

2.14.1 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].

2.14.1.1 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.
	-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.

Commands	Parameters	Function
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.

2.14.1.2 Usage Examples

Function 1: GEF to plain table GEM format

```
$ singularity exec SAW_v5.4.0.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_v5.4.0.sif cellCut view \  ## convert a whole GEF
      -s {SN} \
      -i /path/to/output/05.tissuecut/{SN}.gef \
      -o {SN}.gem
$ singularity exec SAW_v5.4.0.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_v5.4.0.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_v5.4.0.sif cellCut bgef \  ## complete GEF that only contains bin1 geneExp group to a whole GEF, you may specify the bin size you need using “-b”. Separate 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_v5.4.0.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 rapidRegister

SAW **`rapidRegister`** is a lite **`register`** pipeline that performs all modules in the **`register`** except cell segmentation. The usage is the same as **`register`**.

Run **`rapidRegister`** requires the following files:

- **`count`** output gene expression matrix file **(.raw.gef)**
- ImageQC processed microscopic tissue staining image file **(.tar.gz)**
- ImageQC Image process record file **(.ipr, require imageQC version >= 1.1.0)**

Expected running time for ~1G reads: ~20 min, Memory: ~20 G

2.14.2.1 Arguments and Options

Table 2-16 `rapidRegister` Arguments and Options

Parameter	Function
`-i`	(Required) ImageQC processed staining image TAR.GZ file or image pre-processed output directory.
`-c`	(Required) ImageQC 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 **`rapidRegister`** outputs.

2.14.2.2 Usage Examples

The usage of **rapidRegister** is similar to **register**. Here we only take scenario 1 as an example.

Scenario 1: Process images from ImageQC output raw data and gene expression matrix. Perform stitching, tissue 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_v5.4.0.sif rapidRegister \
            -i ${image4register} \
            -c ${imageQC} \
            -v /path/to/output/03.count/{SN}.raw.gef \
            -o /path/to/output/04.register
```

2.14.2.3 Outputs

rapidRegister output files are organized as below:

```
$ tree /path/to/output/04.register
/path/to/output/04.register
...  ## skip setting files, logs and image folder
├── attrs.json
├── fov_stitched_transformed.tif
├── SN_date_time_version.ipr  ## rapidRegister output IPR does not have /CellSeg/
CellMask dataset
├── SN_tissue_bbox.csv
└── transform_thumb.png
```

2.14.3 checkGTF

SAW **checkGTF** is an application for checking whether the GTF/GFF file is in the correct format as the input for **count**.

Run **checkGTF** requires the following files:

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

2.14.3.1 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.

2.14.3.2 Usage Examples

```
$ singularity exec SAW_v5.4.0.sif checkGTF \
      -i /path/to/reference/genes.gtf \
      -o /path/to/reference/genes_new.gtf \
```

2.14.4 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).

2.14.4.1 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 group name 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.
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.
	-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.

2.14.4.2 Usage Examples

Function 1: **img2rpi**

```
$ singularity exec SAW_v5.4.0.sif imageTools img2rpi \
    -i /path/to/output/04.register/fov_stitched_transformed.tif \
    -g ssDNA \
    -b 1 10 50 100 \
    -o /path/to/output/04.register/fov_stitched_transformed.rpi
```

Function 2: **merge**

```
$ singularity exec SAW_v5.4.0.sif imageTools merge \
    -i /path/to/output/04.register/{SN}_regist.tif,/path/to/output/04.register/{SN}_
tissue_cut.tif,/path/to/output/04.register/{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/fov_stitched_transformed.tif
imageIPR=$(find /path/to/output/04.register -maxdepth 1 -name {SN}*.ipr | head -1)
stitchTmplt=/path/to/output/04.register/transform_template.txt

singularity exec SAW_5.4.0.sif imageTools overlay \
    -i ${preRegImage} \
    -c ${imageIPR} \
    -o /path/to/output/04.register/stitch_check.tif \
    -d stitch \
    -l 60 \
    -w 3

singularity exec ${sif} imageTools overlay \
    -i ${preRegImage} \
    -c ${stitchTmplt} \
    -o /path/to/output/04.register/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/{SN}_regist.tif
imageIPR=$(find /path/to/output/04.register -maxdepth 1 -name {SN}*.ipr | head -1)
regTmplt=/path/to/output/04.register/matrix_template.txt

singularity exec SAW_5.4.0.sif imageTools overlay \
    -i ${regImage} \
    -c ${imageIPR} \
    -o /path/to/output/04.register/register_check.tif \
    -d register \
    -l 60 \
    -w 3

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

CHAPTER 3
TEST DATA DEMONSTRATION

Users may refer to this section as a format for testing SAW process, of the files show 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_barcodeMap.stat
...
getBarcodePositionMap_uniqBarcodeTypes: 645784920
total_reads:    1002214171
reads_with_polyA:       131113905       13.08%
reads_filteredByPolyA:  22008148        2.20%
mapped_reads:   826344259       82.45%
reads_with_adapter:     9007116 0.90%
reads_with_dnb: 42264284        4.22%
barcode_exactlyOverlap_reads:   682746301       68.12%
barcode_misOverlap_reads:       143590127       14.33%
barcode_withN_reads:    7831    0.00%
Q10_bases_in_barcode:   99.54%
Q20_bases_in_barcode:   97.49%
Q30_bases_in_barcode:   91.74%
Q10_bases_in_umi:       99.26%
Q20_bases_in_umi:       96.32%
Q30_bases_in_umi:       89.45%
Q10_bases_in_seq:       99.47%
Q20_bases_in_seq:       97.12%
Q30_bases_in_seq:       91.08%
umi_filter_reads:       8265089 0.82%
umi_with_N_reads:       13025   0.00%
umi_with_polyA_reads:   12365   0.00%
umi_with_low_quality_base_reads:        8239699 0.82%
mapped_dnbs: 75619113
...
```

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 |       766807770
                      Average input read length |       95
                                    UNIQUE READS:
                   Uniquely mapped reads number |       643871246
                        Uniquely mapped reads % |       83.97%
                          Average mapped length |       95.21
                       Number of splices: Total |       67595584
            Number of splices: Annotated (sjdb) |       65674308
                       Number of splices: GT/AG |       66407685
                       Number of splices: GC/AG |       457595
                       Number of splices: AT/AC |       41563
               Number of splices: Non-canonical |       688741
                      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 |       87649341
             % of reads mapped to multiple loci |       11.43%
        Number of reads mapped to too many loci |       5301054
             % 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 |       28773993
                 % of reads unmapped: too short |       3.75%
              Number of reads unmapped: other |       1212136
                   % 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.sort-
edByCoord.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.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
731520587       643871246       532386027       108123310       11.98   17.31   79.69
## ANNOTATION METRICS
TOTAL_READS     MAP     EXONIC  INTRONIC        INTERGENIC      TRANSCRIPTOME   ANTISENSE
643871246       643871246       483163052       49222975        111485219       532386027
109940618
100.0   100.0   75.0    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
E100026571L1C003R03702347721    0       1       3001778 255     100M    *       0       0
GTATGACATCTGTCCAGGATCTTCTAGCTTTCATAGTCTCTGGTGAGAAGTCTGGAGTAATTCTAATAGGCCTGCATTTATATGT-
TACTTGACCTTTTTC    EEFEDFFEFFFFEFFFFEC@EFFFFDFFEEFFEFFFFCFCEFFAFBFCED??FGBEFFDC:FFFDCFAF-
4FAFFDFFDG?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
E100026571L1C005R02302788444    528     1       3016331 0       100M    *       0       0
TTTATGTGGAGTTCCTTAATCCACTTAGATTTGACCTTAGTACAAGGAGATAGGAATGGATCAATTCGCATTCTTCTACATGATAACAG-
CCAGTTGTACC    ;FDF>FCFFEAD:FFEBF=@FFDEEFFFC@EFCEFDDFFCE?FDFF7EEECFDEFFFCEFCCEEDEEEFEFB-
FEEFFDEEFFFEEDFFEDFEEEEFFEED    NH:i:5  HI:i:1  AS:i:96 nM:i:1  Cx:i:6628        Cy:i:7872
UR:Z:EDFF9
```

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 { ( 76041339 ) / ( 76041339 ) }
   DATA {
   (0): {
         4888,
         10392,
         1
      },
   (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 { ( 24661 ) / ( 24661 ) }
   DATA {
   (0): {
         “Gm1992”,
         0,
         132
      },
   (1): {
```

3.3.4 Example of count Sampling File

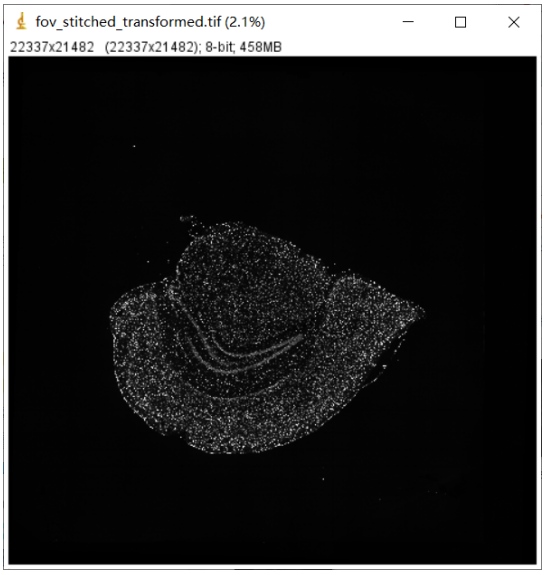
```
$ head -8 /path/to/output/03.count/SS200000135TL_D1_raw_barcode_gene_exp.txt
y x geneIndex MIDIndex readCount
10392 4888 10551 665954 4
7096 8901 10551 881671 1
7096 8901 10551 357383 20
18783 7397 10551 355789 1
13032 9155 10551 297666 1
13032 9155 10551 298690 1
11778 10617 10551 686313 4
```

3.4 register and imageTools

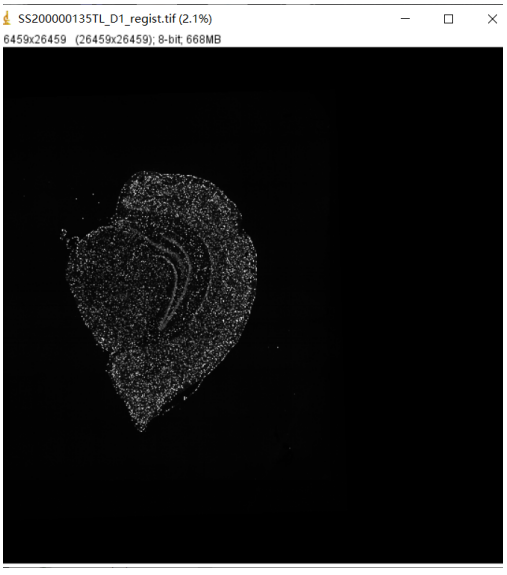
3.4.1 Registered image

File /path/to/output/04.register/fov_stitched_transformed.tif and /path/to/output/04.register/SS200000135TL_D1_regist.tif.

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



/path/to/output/04.register/
SS200000135TL_D1_regist.tif

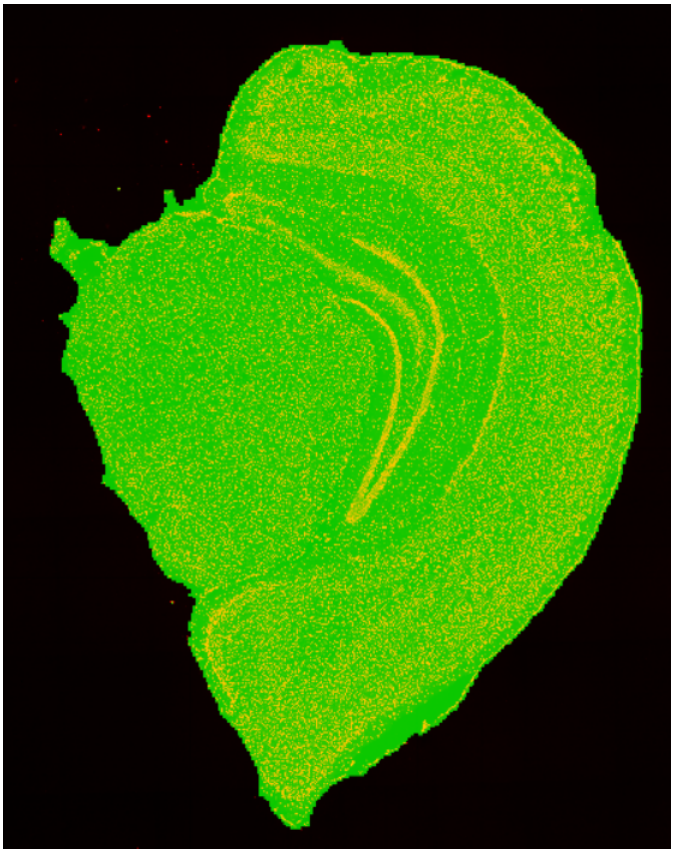


3.4.2 Image Process Record File

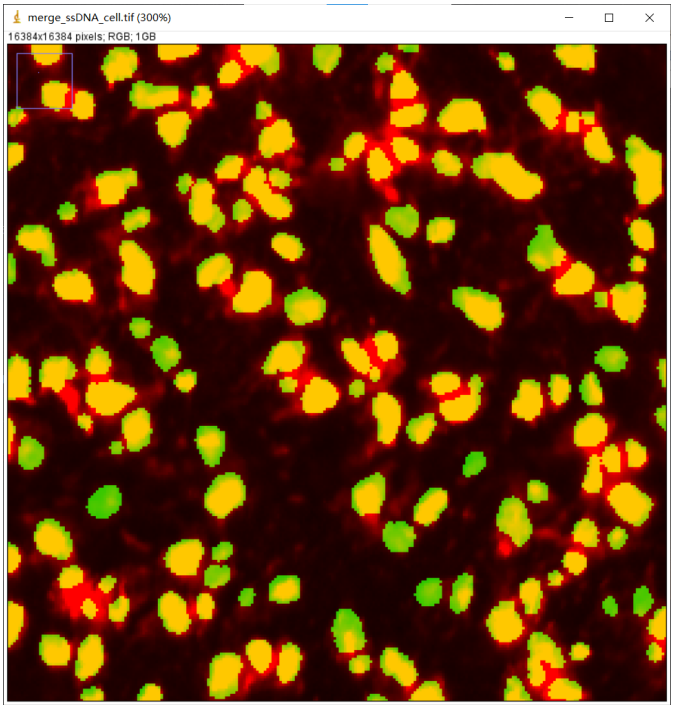
```
h5dump -n /path/to/output/04.register/SS200000135TL_D1_20220527_201353_1.1.0.ipr
HDF5 “/path/to/output/04.register/SS200000135TL_D1_20220527_201353_1.1.0.ipr” {
FILE_CONTENTS {
 group      /
 group      /CellSeg
 dataset    /CellSeg/CellMask
 group      /ImageInfo
 dataset    /ImageInfo/RGBScale
 group      /ManualState
 dataset    /Preview
 group      /QCInfo
 group      /QCInfo/CrossPoints
 dataset    /QCInfo/CrossPoints/0_0
...
 dataset    /QCInfo/CrossPoints/9_7
 dataset    /QCInfo/TrackDistanceTemplate
 group      /Register
 dataset    /Register/MatrixTemplate
 group      /StereoResepSwitch
 group      /Stitch
 group      /Stitch/BGIStitch
 dataset    /Stitch/BGIStitch/StitchedGlobalLoc
 group      /Stitch/ScopeStitch
 dataset    /Stitch/ScopeStitch/GlobalLoc
 group      /Stitch/StitchEval
 dataset    /Stitch/StitchEval/StitchEvalH
 dataset    /Stitch/StitchEval/StitchEvalV
 dataset    /Stitch/TemplatePoint
 dataset    /Stitch/TransformTemplate
 group      /TissueSeg
 dataset    /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): “0.0.1”
      }
   }
   GROUP “CellSeg” {
      ATTRIBUTE “CellSegShape” {
         DATATYPE  H5T_STD_I64LE
         DATASPACE  SIMPLE { ( 2 ) / ( 2 ) }
         DATA {
         (0): 21482, 22337
```

3.4.3 ImageTools merge

Merged image of microscopy image `SS200000135TL_D1_regist.tif` and tissue segmentation mask file `SS200000135TL_D1_tissue_cut.tif` to check tissue segmentation performance.

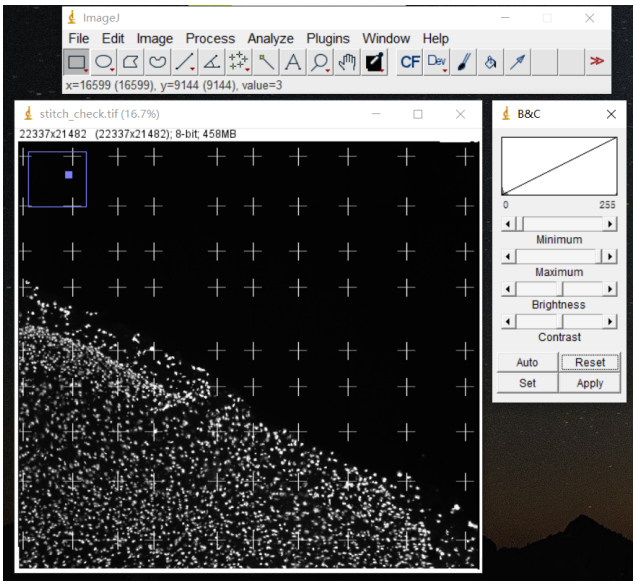


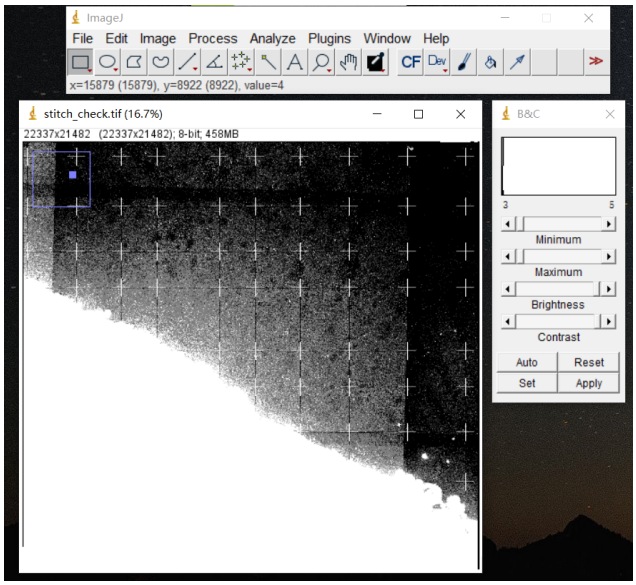
Part of merged image of microscopy image `SS200000135TL_D1_regist.tif` and cell segmentation mask file `SS200000135TL_D1_mask.tif` to check cell segmentation performance.



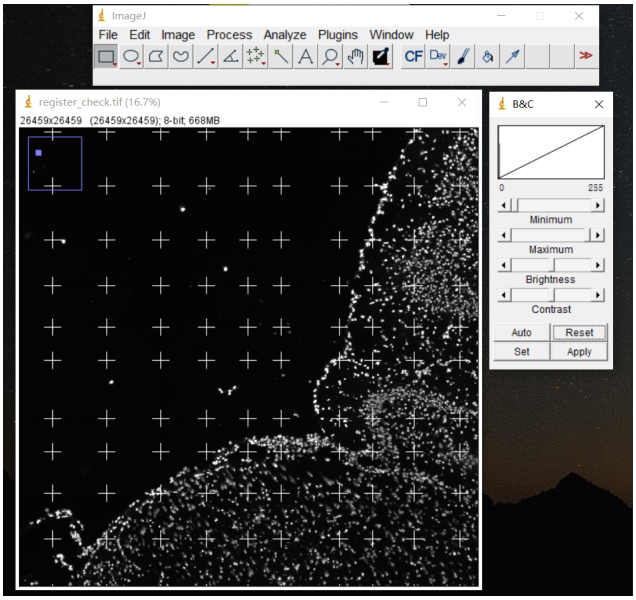
3.4.4 ImageTools overlay

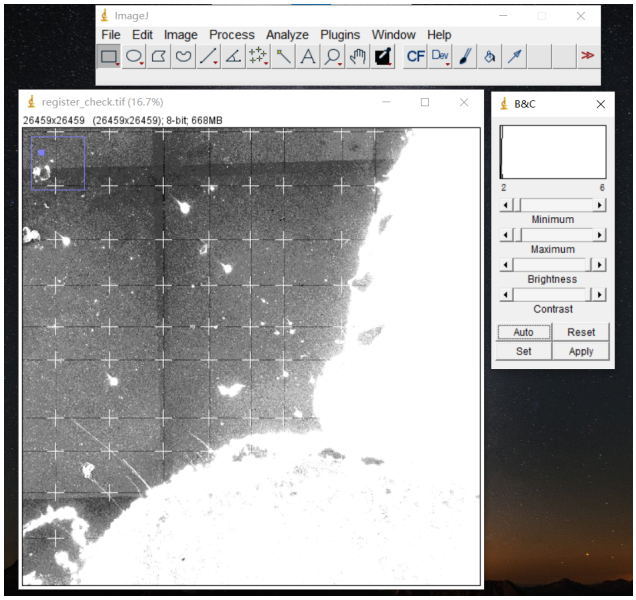
Stack stitching template onto the `fov_stitched_transformed.tif` to check the result of stitching.





Stack registration template onto the `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: 88637560
Number_of_DNB_under_tissue: 36679634
Ratio: 41.38%
Total_gene_type: 24299
MID_counts: 89816137
Fraction_MID_in_spots_under_tissue: 83.07%
Reads_under_tissue: 648371996
Fraction_reads_in_spots_under_tissue: 78.46%

binSize=1
Mean_reads_per_spot: 17.68
Median_reads_per_spot: 11.00
Mean_gene_type_per_spot: 1.71
Median_gene_type_per_spot: 1
Mean_Umi_per_spot: 2.45
Median_Umi_per_spot: 2

binSize=50
Mean_reads_per_spot: 18045.92
Median_reads_per_spot: 16198.00
Mean_gene_type_per_spot: 1151.22
Median_gene_type_per_spot: 1117
Mean_Umi_per_spot: 2499.82
Median_Umi_per_spot: 2309

binSize=100
Mean_reads_per_spot: 71116.81
Median_reads_per_spot: 64454.00
Mean_gene_type_per_spot: 3083.32
Median_gene_type_per_spot: 3081
Mean_Umi_per_spot: 9851.50
Median_Umi_per_spot: 9066

binSize=150
Mean_reads_per_spot: 157601.36
Median_reads_per_spot: 143773.00
Mean_gene_type_per_spot: 4891.22
Median_gene_type_per_spot: 5029
Mean_Umi_per_spot: 21831.83
Median_Umi_per_spot: 20242

binSize=200
Mean_reads_per_spot: 276727.25
Median_reads_per_spot: 254272.00
Mean_gene_type_per_spot: 6403.27
Median_gene_type_per_spot: 6719
Mean_Umi_per_spot: 38333.82
Median_Umi_per_spot: 35679
```

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 { ( 62647604 ) / ( 62647604 ) }
   DATA {
   (0): {
         4888,
         10392,
         1
      },
   (1): {

h5dump -d /geneExp/bin1/gene /path/to/output/05.tissuecut/SS200000135TL_D1.tissue.
gef | head -20
HDF5 “/path/to/output/05.tissuecut/SS200000135TL_D1.tissue.gef” {
DATASET “/geneExp/bin1/gene” {
   DATATYPE  H5T_COMPOUND {
      H5T_STRING {
         STRSIZE 32;
         STRPAD H5T_STR_NULLPAD;
         CSET H5T_CSET_ASCII;
         CTYPE H5T_C_S1;
      } “gene”;
      H5T_STD_U32LE “offset”;
      H5T_STD_U32LE “count”;
   }
   DATASPACE  SIMPLE { ( 24299 ) / ( 24299 ) }
   DATA {
   (0): {
         “Gm1992\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\00
0\000\000\000\000\000\000\000\000\000\000\000\000”,
         0,
         112
      },
   (1): {
```

3.5.3 Example of Gene Expression Matrix for Maximum Area Enclosing Rectangle

```
$ h5dump -n /path/to/output/05.tissuecut/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/05.tissuecut/SS200000135TL_D1.gef | head -20
HDF5 “/path/to/output/05.tissuecut/SS200000135TL_D1.gef” {
DATASET “/stat/gene” {
   DATATYPE  H5T_COMPOUND {
      H5T_STRING {
```

```
         STRSIZE 32;
         STRPAD H5T_STR_NULLTERM;
         CSET H5T_CSET_ASCII;
         CTYPE H5T_C_S1;
      } “gene”;
      H5T_STD_U32LE “MIDcount”;
      H5T_IEEE_F32LE “E10”;
   }
   DATASPACE  SIMPLE { ( 24661 ) / ( 24661 ) }
   DATA {
   (0): {
         “Gm42418”,
         5861037,
         60.1033
      },
   (1): {
```

3.6 cellCut

3.6.1 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

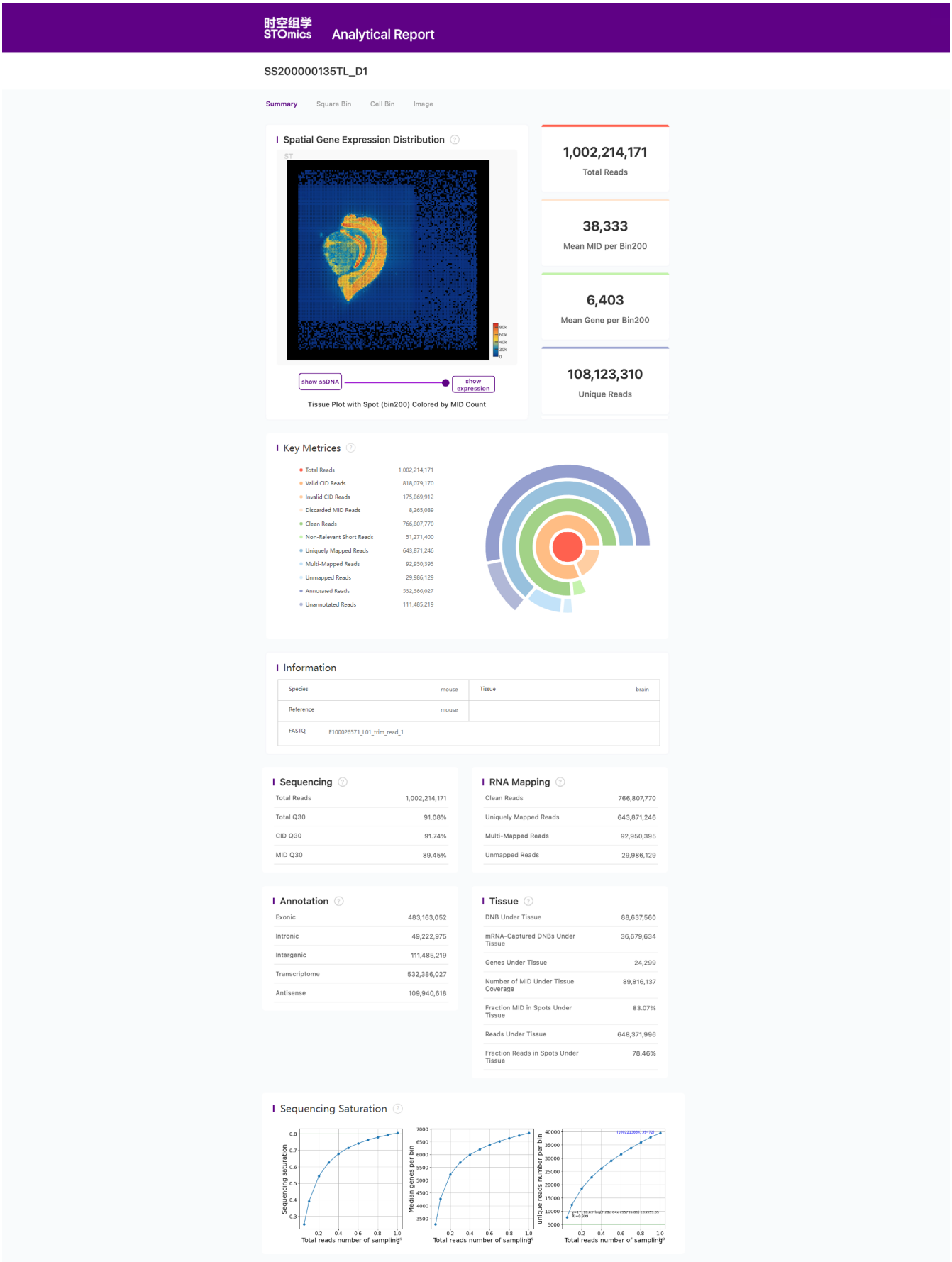
```
$ cat /path/to/output/07.saturation/sequence_saturation.tsv
sample  bar_x   bar_y1  bar_y2  bar_umi bin_x   bin_y1  bin_y2  bin_umi
0.05    26619302        0.250959        1       19938952        26619302
0.27571 3270    7613
0.1     53238604        0.390241        1       32462699        53238604
0.41122 4268    12394
0.2     106477208       0.543149        1       48644210        106477208
0.558617        5215    18573
0.3     159715808       0.625887        1       59751787        159715808
0.638094        5693    22814
0.4     212954416       0.67839 1       68488171        212954416
0.688522        5995    26150
0.5     266193008       0.714813        1       75914701        266193008
0.723539        6204    28985
0.6     319431616       0.741736        1       82497808        319431616
0.749427        6378    31499
0.7     372670208       0.76249 1       88513055        372670208
0.769402        6517    33795
0.8     425908832       0.779116        1       94076279        425908832
0.78542 6642    35920
0.9     479147392       0.792733        1       99311385        479147392
0.798541        6747    37918
1       532386027       0.804159        1       104262941       532386027
0.809561        6840    39472
```

3.8 report

3.8.1 Example of Statistical Summary Report

```
cat /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”,
                “mapped_reads”: “826344259(82.45%)”,
                “CID_misOverlap_reads”: “143590127(14.33%)”,
```

3.8.2 HTML Report



时空组学 STOmics Analytical Report

SS200000135TL_D1

Summary | Square Bin | Cell Bin | Image

Spatial Gene Expression Distribution

show ssDNA — show expression

Tissue Plot with Spot (bin200) Colored by MID Count

1,002,214,171 Total Reads

38,333 Mean MID per Bin200

6,403 Mean Gene per Bin200

108,123,310 Unique Reads

Key Metrices

Total Reads	1,002,214,171
Valid CID Reads	818,079,170
Invalid CID Reads	175,869,912
Discarded MID Reads	8,265,089
Clean Reads	766,807,770
Non-Relevant Short Reads	51,271,400
Uniquely Mapped Reads	643,871,246
Multi-Mapped Reads	92,950,395
Unmapped Reads	29,986,129
Annotated Reads	532,386,027
Unannotated Reads	111,485,219

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	766,807,770
Uniquely Mapped Reads	643,871,246
Multi-Mapped Reads	92,950,395
Unmapped Reads	29,986,129

Annotation

Exonic	483,163,052
Intronic	49,222,975
Intergenic	111,485,219
Transcriptome	532,386,027
Antisense	109,940,618

Tissue

DNB Under Tissue	88,637,560
mRNA-Captured DNBs Under Tissue	36,679,634
Genes Under Tissue	24,299
Number of MID Under Tissue Coverage	89,816,137
Fraction MID in Spots Under Tissue	83.07%
Reads Under Tissue	648,371,996
Fraction Reads in Spots Under Tissue	78.46%

Sequencing Saturation

Sequencing saturation — Median genes per bin — unique reads number per bin

Total reads number of sampling

时空组学 STOmics Analytical Report

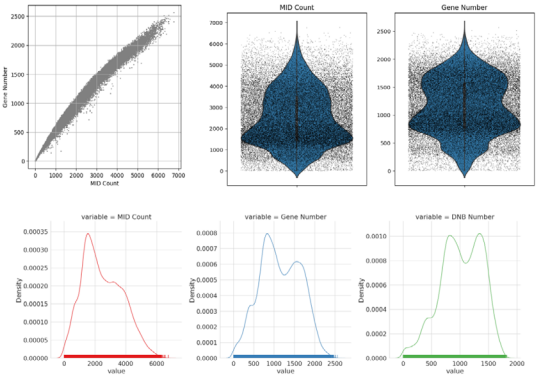
SS200000135TL_D1

Summary Square Bin Cell Bin Image

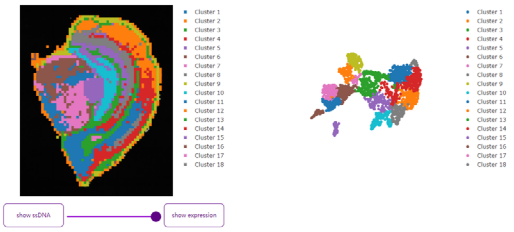
Tissue Square Bin Statistics

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)
50	18,045	16,198	1,151	1,117	2,499	2,309
100	71,116	64,454	3,083	3,081	9,851	9,066
150	157,601	143,773	4,891	5,029	21,831	20,242
200	276,727	254,272	6,403	6,719	38,333	35,679

Bin 50



Clustering



Tissue Plot with Spots (bin200)

UMAP Projection of Spots (bin200)

时空组学 STOmics Analytical Report

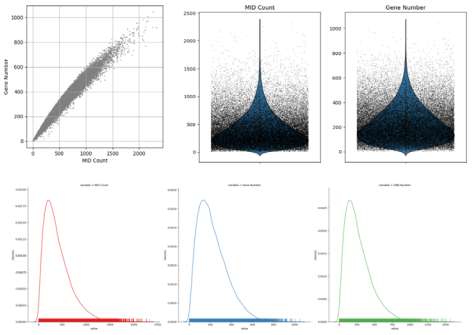
SS200000135TL_D1

Summary Square Bin Cell Bin Image

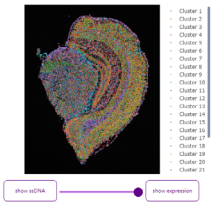
Cell Bin Statistics

Mean Cell Area	278	Median Cell Area	264
Mean DNB Count	273	Median DNB Count	231
Mean Gene Type	223	Median Gene Type	196
Mean MID	398	Median MID	334

Cell Bin



Clustering



Tissue Plot with Spots (cell bin)

UMAP Projection of Spots (cellbin)

时空组学 STOmics Analytical Report

SS200000135TL_D1

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_20220527_201353_1.1.0

QC

ImageQC Version	1.1.0
QC Pass	Pass
Track Line Score	70
Clarity Score	94
Good FOV Count	92
Total FOV Count	117
Stitching Score	93
Tissue Segmentation Score	100
Registration Score	99

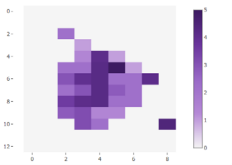
Stitching

Template Source Row No.	9
Template Source Column No.	6
Global Height	24,071
Global Width	25,128

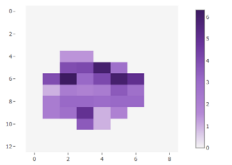
Registration

ScaleX	0.866
ScaleY	0.866
Rotation	-1.603
Flip	True
Image X Offset	-3,383.29
Image Y Offset	2,065
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/BGIResearch/SAW).

```
$ tree
.
|-- image
|   |-- SS200000135TL_D1_20220527_201353_1.1.0.ipr
|   `-- SS200000135TL_D1_20220527_201353_1.1.0.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
    |   |-- Genome
    |   |-- SA
    |   |-- SAindex
    |   |-- chrLength.txt
    |   |-- chrName.txt
    |   |-- chrNameLength.txt
    |   |-- chrStart.txt
    |   |-- exonGeTrInfo.tab
    |   |-- exonInfo.tab
    |   |-- geneInfo.tab
    |   |-- genomeParameters.txt
    |   |-- genomeParameters_bkp.txt
    |   |-- sjdbInfo.txt
    |   |-- sjdbList.fromGTF.out.tab
    |   |-- sjdbList.out.tab
    |   `-- transcriptInfo.tab
    |-- genes.gtf
    `-- genome.fa

5 directories, 24 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.Aligned.sortedByCoord.out.bam.csi	BAM index file, which is a table of contents for the BAM file that indicating where in the BAM file a specific read of set of reads can be found.
	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_barcodeMap.stat	Statistical report of CID mapping, such as CID mapped reads count, reads sequencing quality, mapped DNB count, etc.
MERGE	SN.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 stands 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.
register& ipr2img	date-hh-mm-ss.log	Log file.
	attrs.json	Height and width of gene expression matrix.
	SN or other user specified name for the image folder used when input into imageQC	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.

STEP	OUTPUT	FILE DESCRIPTION
REGISTER& IPR2IMG	fov_stitched_transformed.tif	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.
	matrix_template.txt	Track line cross point template for the registered ssDNA image. Used for assessing registeration result.
	SN_date_time_version.ipr	IPR format image process record file which records basic image information gathered from imageQC.
	SN_mask.tif	Registered cell segmentation binary image file in TIFF format.
	SN_regist.tif	Registered panoramic image file in TIFF format.
	SN.rpi	Registered panoramic image file, tissue area boundary, and cell boundaries (downsampled) that stores as an image pyramid.
	SN_tissue_bbox.csv	Registered panoramic image size.
	SN_tissue_cut.tif	The tissue segmentation result of registered panoramic image file in TIFF format.
	transform_template.txt	Track line cross point template for the fov_stitched_transformed.tif. Used for assessing stitching result.
	transform_thumb.png	A downsampled pre-registered panoramic image.
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 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.
	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.
	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 number in each bin (bin100)
	scatter_150x150_MID_gene_counts.png	Scatter plot of MID count and gene number in each bin (bin150)
	scatter_200x200_MID_gene_counts.png	Scatter plot of MID count and gene number in each bin (bin200)
	scatter_50x50_MID_gene_counts.png	Scatter plot of MID count and gene number in each bin (bin50)
	statistic_100x100_MID_gene_DNB.png	Univariate distribution of MID count, gene numbers, and DNB numbers with rug along the x axis (bin100)
	statistic_150x150_MID_gene_DNB.png	Univariate distribution of MID count, gene numbers, and DNB numbers with rug along the x axis (bin150)
	statistic_200x200_MID_gene_DNB.png	Univariate distribution of MID count, gene numbers, and DNB numbers with rug along the x axis (bin200)
	statistic_50x50_MID_gene_DNB.png	Univariate distribution of MID count, gene numbers, 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 number in each bin (bin100)
	violin_150x150_MID_gene.png	Violin plots show the distribution of deduplicated MID count and gene number in each bin (bin150)

STEP	OUTPUT	FILE DESCRIPTION
TISSUECUT	violin_200x200_MID_gene.png	Violin plots show the distribution of deduplicated MID count and gene number in each bin (bin200)
	violin_50x50_MID_gene.png	Violin plots show the distribution of deduplicated MID count and gene number in each bin (bin50)
	tissuecut.stat	Statistical report for tissue 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 record by its boundary. The origin of expression matrix has been calibrated to (0,0), and the coordinate offset x and y has been recorded as offsetX and offsetY in the attribute of the GEF file same to minX and minY in the raw GEF file.
SPATIAL-CLUSTER	SN.spatial.cluster.h5ad	H5AD file for spatial clustering analysis result.
CELLCLUSTER	SN.cell.cluster.h5ad	H5AD file for cell clustering analysis result.
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 number in each cell
	statistic_1x1_MID_gene_DNB.png	Univariate distribution of MID count, gene numbers, and DNB numbers with rug along the x axis of each cell
	violin_1x1_MID_gene.png	Violin plots show the distribution of deduplicated MID count and gene number 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.

imageTools

- **`OSError` raises for `imageTools` merge**

```
OSError: [Errno 30] Read-only file system: '/opt/saw_v5.4.0_software/pipeline/image-Tools/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`.

References

1. BGIResearch/SAW. Accessed October 13, 2021. https://github.com/BGIResearch/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. 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.*; 2021. 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. GitHub - BGIResearch/stereopy: A toolkit of spatial transcriptomic analysis. Accessed July 4, 2021. https://github.com/BGIResearch/stereopy
12. BGIResearch/geftools: Tools for manipulating GEFs. Accessed April 7, 2022. https://github.com/BGIResearch/geftools
13. BGIResearch/gefpy: gef io, draw out from stereopy. Accessed April 7, 2022. https://github.com/BGIResearch/gefpy

Contact Us

BGI-Research, Shenzhen

https://www.stomics.tech/EN/

Email: support@stereomics.com

Please raise GitHub issues for reporting bugs and requesting features:

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