时空组学
STOmics

STEREO-SEQ ANALYSIS WORKFLOW SOFTWARE SUITE

USER MANUAL

Software Version: V5.1.3

Manual Version: A3

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;

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

©2022 Beijing Genomics Institute at Shenzhen (BGI-Research). All rights reserved.

1. The products shall be for research use only, not for use in diagnostic.

2. The Content on this manual may be protected in whole or in part by applicable intellectual property laws. BGI-Research and / or corresponding right subjects own their intellectual property rights according to law, including but not limited to trademark rights, copyrights, etc.

3. BGI-Research do not grant or imply the right or license to use any copyrighted content or trademark (registered or unregistered) of us or any third party. Without our written consent, no one shall use, modify, copy, publicly disseminate, change, distribute, or publish the program or Content of this manual without authorization, and shall not use the design or use the design skills to use or take possession of the trademarks, the logo or other proprietary information (including images, text, web design or form) of us or our affiliates.

4. Nothing contained herein is intended to or shall be construed as any warranty, expression or implication of the performance of any products listed or described herein. Any and all warranties applicable to any products listed herein are set forth in the applicable terms and conditions of sale accompanying the purchase of such product. BGI-Research, Shenzhen makes no warranty and hereby disclaims any and all warranties as to the use of any third-party products or protocols described herein.

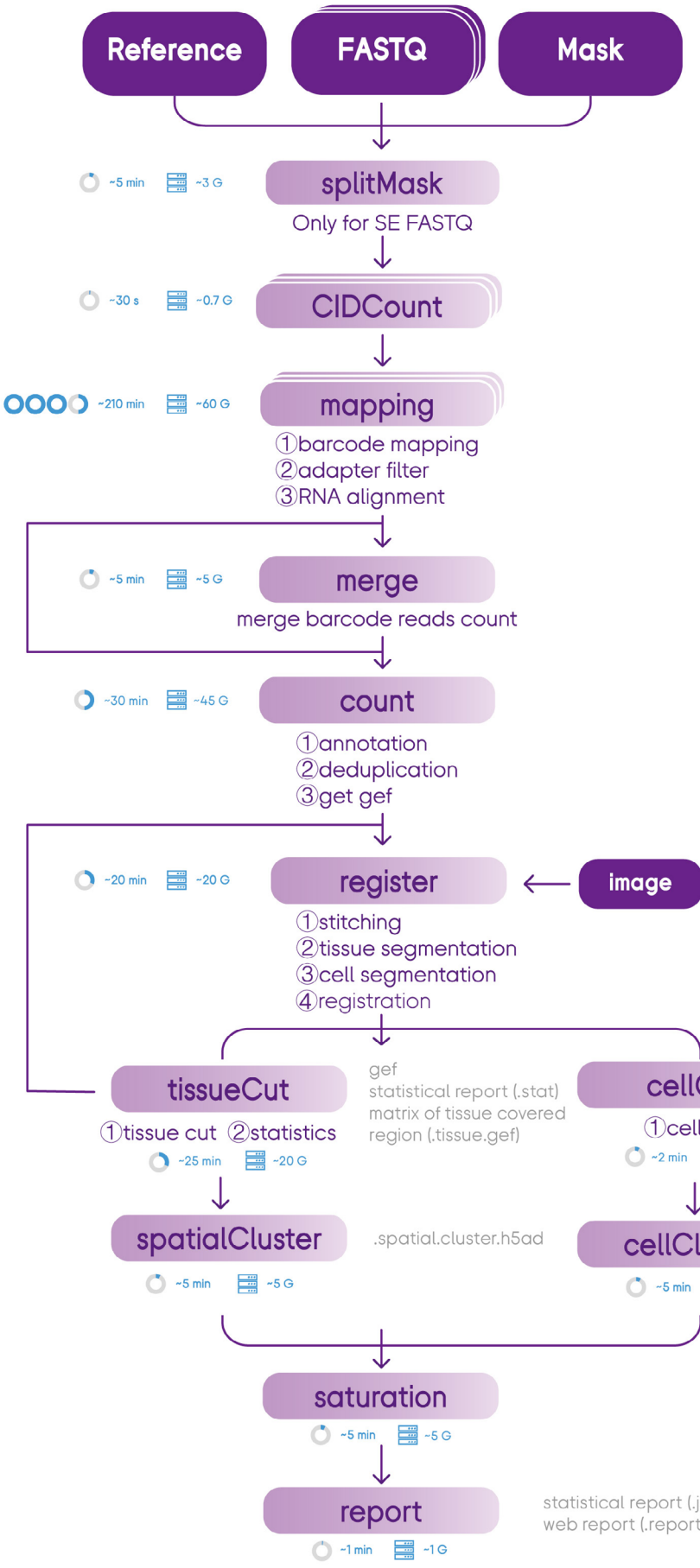
WORKFLOW

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

TABLE OF CONTENTS

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 suggest 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 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 mapping of 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.
- **ipr2img:** 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 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`.

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
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

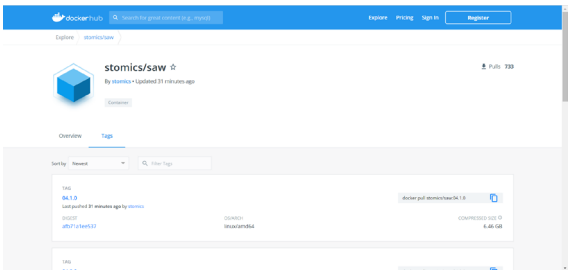
SAW >= 5.0.0 requires imageQC version >= 1.1.0 that provide IPR file for recording image processing data.

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 5.1.3 as an example.

1.4.1. SAW Installation via Singularity

CRITICAL STEPS: For yellow highlighted inputs, please replace with your own path or data.

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

```
$ singularity build SAW_v5.1.3.sif docker://stomics/saw:05.1.3  ## option 1
$ singularity build --sandbox SAW_v5.1.3/ docker://stomics/saw:05.1.3  ## 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_v5.1.3/ docker://stomics/saw:05.1.3
```

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_v5.1.3.sif <application>  ## option 1.1
$ singularity exec /path/to/SAW_v5.1.3.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_v5.1.3.sif /bin/bash  ## option 2.1
Singularity>
Singularity> <shell-command>
Singularity> exit
$
$ singularity shell /path/to/SAW_v5.1.3.sif  ## option 2.2
Singularity>
Singularity> <shell-command>
Singularity> exit
$
$ singularity shell SAW_v5.1.3  ## 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_v5.1.3.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:05.1.3
```

Step 2: Run pipelines interactively

```
$ docker run -it stomics/saw:05.1.3
```

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),

```
$ mkdir /path/to/output/00.splitmask
$ singularity exec SAW_v5.1.3.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 small program for computing the number of CIDs in the Stereo-seq Chip T Mask file and roughly estimating how much memory will be needed to do **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_v5.1.3.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.

```
$ singularity exec SAW_v5.1.3.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}"
```

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

2.2.3 Outputs

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

```
$ singularity exec SAW_v5.1.3.sif CIDCount -i SN.barcodeToPos.h5 -s mouse -g 3
645784920  ## CID count
62  ## estimated memory for mapping
```

If users wish to run **CIDCount**, they may add "**bcNum**" to the required **{lane}.bcPara** file for **mapping**. Please check **2.3.2 Usage Example (mapping)** for the demo.

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 slice. **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 STOmics Chip Mask file. Coordinate information for reads that CID could be paired with will be added based on the records of the 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. Reads that get the coordinate annotations are Valid CID mRNA Reads (**Valid CID Reads**). After discarding reads with MID quality does 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)**
- STOmics 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.
`--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 2-2.
`--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. If PE sequencing, then specify the path to the FASTQ file of read1 here.
`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: **SEQ500**, **T1**, or **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 STOmicsd Chip T serial number (SN, e.g. SS200000135TL_D1), 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 if run ahead mapping
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 if run ahead mapping
polyAnum=15
mismatchInPolyA=2
```

Run **mapping** for PE input:

```
$ singularity exec SAW_v5.1.3.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/00.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 if run ahead mapping
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 if run ahead mapping
polyAnum=15
mismatchInPolyA=2
```

Run **mapping** pipeline

```
$ singularity exec SAW_v5.1.3.sif mapping \
                --outSAMattributes spatial \
                --outSAMtype BAM SortedByCoordinate \
                --genomeDir /path/to/genomeDir \
                --runThreadN 8 \
                --outFileNamePrefix /path/to/output/01.mapping/{index}. \  ## {index} or {idx} depending on the scenario
                --sysShell /bin/bash \
                --stParaFile /path/to/output/01.mapping/{index}.bcPara \  ## {index} or {idx} depending on the scenario
                --readNameSeparator \" \" \
                --limitBAMsortRAM 38582880124 \
                --limitOutSJcollapsed 10000000 \
                --limitIObufferSize=280000000 \
                --outBAMsortingBinsN 50 \
                > /path/to/output/01.mapping/{index}_barcodeMap.stat  ## {index} or {idx} depending on the scenario
```

2.3.3 Outputs

PE scenario 1 output files are organized as below:

```
$ ll -Rth /path/to/output/01.mapping
-rw-rw-r--  1 ubuntu ubuntu  11M Apr 13 20:57 lane.Aligned.sortedByCoord.out.bam.csi
-rw-rw-r--  1 ubuntu ubuntu 8.9K Apr 13 20:40 lane.Log.out
-rw-rw-r--  1 ubuntu ubuntu 2.0K Apr 13 20:40 lane.Log.final.out
-rw-rw-r--  1 ubuntu ubuntu  17K Apr 13 20:40 lane.Log.progress.out
-rw-rw-r--  1 ubuntu ubuntu  62G Apr 13 20:40 lane.Aligned.sortedByCoord.out.bam
-rw-rw-r--  1 ubuntu ubuntu 1.1K Apr 13 20:06 lane_barcodeMap.stat
-rw-rw-r--  1 ubuntu ubuntu  11M Apr 13 20:06 lane.SJ.out.tab
-rw-rw-r--  1 ubuntu ubuntu 997M Apr 13 20:06 lane.barcodeReadsCount.txt
-rw-rw-r--  1 ubuntu ubuntu  528 Apr 13 17:46 lane.bcPara
```

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 files:

- **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.

2.2.2 Usage Example

```
$ mkdir /path/to/multi_lane_output/02.merge
$ singularity exec SAW_v5.1.3.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:

```
$ ll -Rth /path/to/multi_lane_output/02.merge
-rw-rw-r--  1 ubuntu ubuntu 997M Apr 13 20:53 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 ST v5, `count` added a mode that allows utilizing 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 **(.gtf / .gff)**

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_v5.1.3.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_v5.1.3.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_v5.1.3.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:

```
$ ll -Rth /path/to/output/03.count
-rw-rw-r--  1 ubuntu ubuntu 636M Apr 13 22:11 SN.raw.gef
-rw-rw-r--  1 ubuntu ubuntu 2.6G Apr 13 22:11 SN_raw_barcode_gene_exp.txt
-rw-rw-r--  1 ubuntu ubuntu  393 Apr 13 22:11 SN.Aligned.sortedByCoord.out.merge.
q10.dedup.target.bam.summary.stat
-rw-rw-r--  1 ubuntu ubuntu  46G Apr 13 22:11 SN.Aligned.sortedByCoord.out.merge.
q10.dedup.target.bam
```

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 expression matrix, and at the same time registered masks from segmentation modules with gene expression matrix using the same parameters.

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 information report **(.json)**

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

2.6.1 Arguments and Options

Table 2-7 `register` Arguments and Options

Parameter	Function
-i	(Required) ImageQC processed staining image TAR.GZ file.
-c	(Required) ImageQC IPR (image process record) file. IPR is designed to efficiently hold images and processing 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	(Required) `count` output gene expression matrix GEF file.
-o	(Required) Path to the directory where to store the `register` outputs.

2.6.2 Usage Example

```
$ 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_v5.1.3.sif register \
                    -i ${image4register} \
                    -c ${imageQC} \
                    -v /path/to/output/03.count/{SN}.raw.gef \
                    -o /path/to/output/04.register
```

2.6.3 Outputs

`register` output files are organized as below:

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

2.7 ipr2img

SAW **`ipr2img`** pipeline (available sine SAW ST >= v5.0.0) is run to “decode” images from the processed IPR file. SAW **`ipr2img`** could output TIFF images from IPR such as pre-registered ssDNA stitched image, tissue segmentation and cell segmentation mask TIFFs, as well as registered these three types of images.

Run **`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 `ipr2img` Arguments and Options

Parameter	Function
`-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.

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_v5.1.3.sif ipr2img \
      -i ${image4register} \
      -c ${imageIPR} \
      -d tissue cell \  ## output both tissue segmentation mask TIFF
      -r True \
      -o /path/to/output/04.register
```

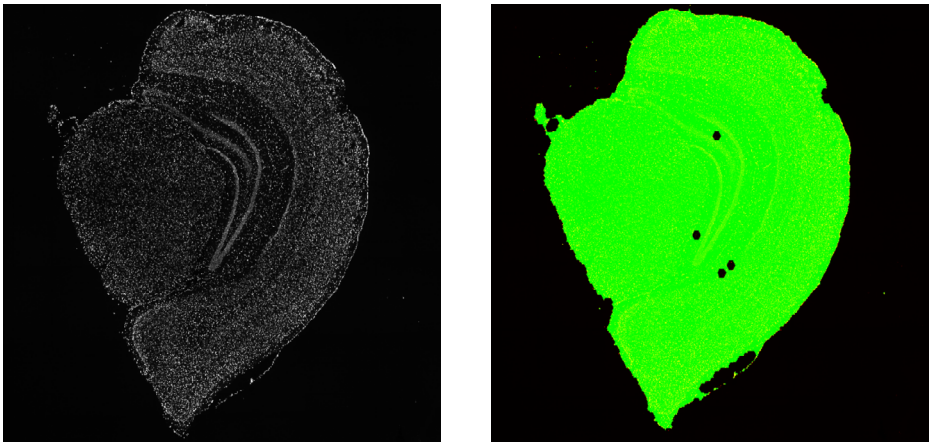
2.7.3 Outputs

`Ipr2img` output files are organized as below:

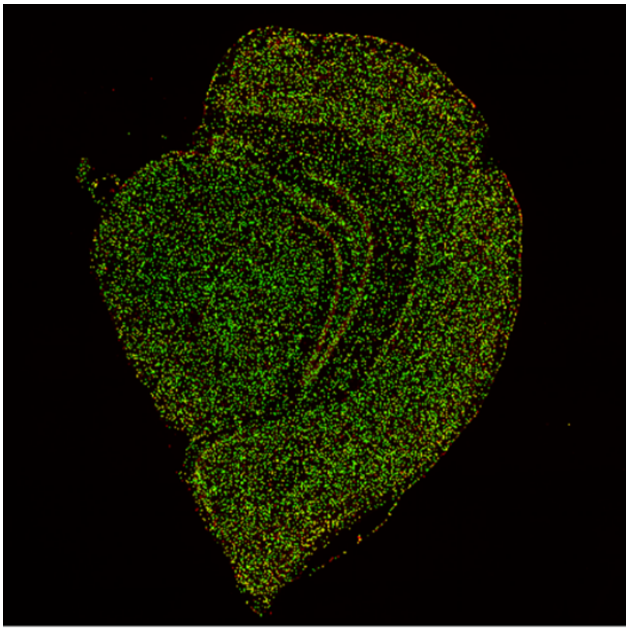
```
$ tree /path/to/output/04.register
/path/to/output/04.register
...  ## skip setting files, logs and image folder
├── attrs.json
├── fov_stitched_transformed.tif
├── matrix_template.txt
├── SN_date_time_version.csv
├── SN_date_time_version.ipr
├── SN_date_time_version.png
├── SN_mask.tif
├── SN_regist.tif
├── SN.rpi
├── SN_tissue_bbox.csv
├── SN_tissue_cut.tif
├── transform_template.txt
└── transform_thumb.png
```

You could pause here and check your tissue segmentation as well as cell segmentation using merged function in ImageJ (images shown below).

Merged image of **`SN_regist.tif`** and **`SN_tissue_cut.tif`** to check tissue segmentation performance.



Merged image of **`SN_regist.tif`** and **`SN_mask.tif`** to check cell segmentation performance



Please refer to **Appendix D: Manual Image Merge Using ImageJ** for further instructions.

2.8 tissueCut

SAW **tissueCut** pipeline could delineate and extract the tissue coverage area based on the aligned image generated from **register** and **ipr2img** 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 python package 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.

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.
-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 directory where the register outputs are stored. Only valid when register has performed.
-t	(Required) Run for tissue segmentation. Valid options: tissue.
--snId	(Required) Stereo-seq Chip T serial number (SN).
--omics	(Required; default to Transcriptomics) String that specify 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:

```
$ mkdir -p /path/to/output/05.tissuecut
$ singularity exec SAW_v5.1.3.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 /path/to/output/04.register \
        -t tissue \
        --snId {SN} \
        --omics=Transcriptomics \
        -d
```

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

```
$ mkdir -p /path/to/output/05.tissuecut
$ singularity exec SAW_v5.1.3.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 \
        -t tissue \
        --snId {SN} \
        --omics=Transcriptomics \
        -d
```

2.8.3 Outputs

`tissueCut` output files:

`Image` is provided:

```
$ tree /path/to/output/05.tissuecut
/path/to/output/05.tissuecut
├── dnb_merge
│   └── bin200.png
├── SN.gef
├── 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
    ├── SN.ssDNA.rpi
    ├── 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
├── dnb_merge
│   └── bin200.png
├── SN.gef
├── 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_v4.1.0.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 cells based on the aligned image generated from **register. cellCut** outputs expression data in cell bin GEF format.

Run **cellCut** requires the following files:

- **count** output gene expression matrix file **(.raw.gef)**
- **register** 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_v5.1.3.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 in the smiliar procedure as **`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) count output 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_v5.1.3.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. Have to be an exist 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_v5.1.3.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_v5.1.3.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
- **count** output statistical report of annotation
- **register** processed Image process record file **(.ipr)**
- **tissueCut** and **cellCut** (if available) output GEF file, statistical report of tissue-covered region, plots and image pyramid RPI file (if available)
- **spatialCluster** and **cellCluster** (if available) output clustering H5AD file
- **saturation** output bin1 and bin200 sequence saturation plot
- 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 have wholeExp/bin200.
-b	(Required) tissueCut output bin 200 scatter plot.
-v	(Required) tissueCut output bin 200 violin plot.
-c	(Required) tissueCut output bin 200 statistics plot.
--bin50Saturation	(Required) tissueCut output bin 50 scatter plot.
--bin50violin	(Required) tissueCut output bin 50 violin plot.
--bin50MIDGeneDNB	(Required) tissueCut output bin 50 statistics plot.
--bin100Saturation	(Required) tissueCut output bin 100 scatter plot.
--bin100violin	(Required) tissueCut output bin 100 violin plot.
--bin100MIDGeneDNB	(Required) tissueCut output bin 100 statistics plot.
--bin150Saturation	(Required) tissueCut output bin 150 scatter plot.
--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.

Parameter	Function
--bin1Saturation	(Required) `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.
-p	(Required) The plotly JS package file path.
--species	(Required) A string of species name.
--tissue	(Required) A string of tissue type.
--logo	(Required) Path of Logo.png.
-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) ImageQC 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 {species_name}, {tissue_type}, {reference_index} with the real information.

```
$ mkdir -p /path/to/output/08.report
$ imageIPR=$(find /path/to/output/04.register -maxdepth 1 -name {SN}*.ipr | head -1)
## has to be a processed IPR
$ singularity exec SAW_v5.1.3.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 \
```

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

```
      --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/05.tissuecut/tissue_fig/{SN}.ssDNA.rpi \
      -s {SN} \
      --pipelineVersion SAW_v5.1.3 \
      -p /opt/saw_v5.1.3_software/pipeline/report_v2.0.2/plotly_package.txt \
      --iprFile ${imageIPR} \
      --species {species_name} \
      --tissue {tissue_type} \
      --reference {reference_index} \
      --logo /opt/saw_v5.1.3_software/pipeline/report_v2.0.2/logo.png \
      -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_v5.1.3.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/scat-
ter_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/statis-
tic_50x50_MID_gene_DNB.png \
      --bin100Saturation /path/to/multi_lane_output/05.tissuecut/tissue_fig/scat-
ter_100x100_MID_gene_counts.png \
      --bin100violin /path/to/multi_lane_output/05.tissuecut/tissue_fig/vio-
lin_100x100_MID_gene.png \
      --bin100MIDGeneDNB /path/to/multi_lane_output/05.tissuecut/tissue_fig/statis-
tic_100x100_MID_gene_DNB.png \
      --bin150Saturation /path/to/multi_lane_output/05.tissuecut/tissue_fig/scat-
ter_150x150_MID_gene_counts.png \
      --bin150violin /path/to/multi_lane_output/05.tissuecut/tissue_fig/vio-
lin_150x150_MID_gene.png \
      --bin150MIDGeneDNB /path/to/multi_lane_output/05.tissuecut/tissue_fig/statis-
tic_150x150_MID_gene_DNB.png \
      -r standard_version \
      -i /path/to/multi_lane_output/05.tissuecut/tissue_fig/{SN}.ssDNA.rpi \
      -s {SN} \
      --pipelineVersion SAW_v5.1.3 \
      -p /opt/saw_v5.1.3_software/pipeline/report_v2.0.2/plotly_package.txt \
      --iprFile ${imageIPR} \
      --species {species_name} \
      --tissue {tissue_type} \
      --reference {reference_index} \
      --logo /opt/saw_v5.1.3_software/pipeline/report_v2.0.2/logo.png
      -o /path/to/multi_lane_output/08.report
```

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

Run **report** 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_v5.1.3.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 \
      -r standard_version \
      -s {SN} \
      --pipelineVersion SAW_v5.1.3 \
      -p /opt/saw_v5.1.3_software/pipeline/report_v2.0.2/plotly_package.txt \
      --species {species_name} \
      --tissue {tissue_type} \
      --reference {reference_index} \
      --logo /opt/saw_v5.1.3_software/pipeline/report_v2.0.2/logo.png \
      -o /path/to/output/08.report
```

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

2.13.3 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 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].

2.13.1 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.
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_v5.1.3.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_v5.1.3.sif cellCut view \  ## convert a whole GEF
      -s {SN} \
      -i /path/to/output/05.tissuecut/{SN}.gef \
      -o {SN}.gem
$ singularity exec SAW_v5.1.3.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_v5.1.3.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_v5.1.3.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_v5.1.3.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.tissue/{SN}.tissue.gef',bin_size=200,n_
thread=4)
>>> bgef.to_gem('{SN}.tissue.bin200.gem')
```

2.14.2 rapidRgister

SAW **rapidRegister** is a lite **register** pipeline that perform all modules in the **register** except cell segmentation.

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.
-c	(Required) ImageQC IPR (image process record) file. IPR is designed to efficiently hold images and processing 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	(Required) 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

Function 1: GEF to plain table GEM format

```
$ 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_v5.1.3.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
├── fov_stitched_transformed.tif
├── SN_date_time_version.csv
├── SN_date_time_version.ipr  ## rapidRegister output IPR does not have /CellSeg/
CellMask dataset
└── SN_date_time_version.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 **(.gtf / .gff)**

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_v5.1.3.sif rapidRegister \
      -i /path/to/reference/genes.gtf \
      -o /path/to/reference/genes_new.gtf \
```

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

"…" iin 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_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.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.Aligned.sorted-
ByCoord.out.bam | head -2
E100026571L1C007R00303973559    256     1       3000644 3       100M    *       0
0       GCCTCATTGTGCCCCATATGTTTGCCTATGTTGTGGACTTATTTTCATTAAACTTTAAAACATCTTTA-
ATTTTTTTCTTTATTTCATCATTGACCAAGCT    -FCA9D?GFFD<-D<CGFEGD-DG*FGFDFBE;E(9BGGE38FFFG-
9GG;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:2Cx:i:8839         Cy:i:7539       UR:Z:120CF
E100026571L1C003R03702347721    0       1       3001778 255     100M    *       0
0       GTATGACATCTGTCCAGGATCTTCTAGCTTTCATAGTCTCTGGTGAGAAGTCTGGAGTAATTCTAATAGGCCTG-
CATTTATATGTTACTTGACCTTTTTC    EEFEDFFEFFFFEFFFFEC@EFFFFDFFEEFFEFFFFCFCEFFAFBFCED??FG-
BEFFDC:FFFDCFAF4FAFFDFFDG?DFBD.F@FECA/FEDEFFAA    NH:i:1  HI:i:1  AS:i:92 nM:i:3Cx-
: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
0       2960    1
0       2972    1
0       2979    1
0       2989    1
0       3057    1
0       3080    1
0       3695    1
0       3703    1
0       3733    1
0       3734    8
```

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 | grep GE:Z | head -2
E100026571L1C002R00703943265    1040    1       3082766 255     11M132671N89M   *       0
0       CTGCTGCAGCTTTTTTTCTTTGAGATTTATTTTTATGCTATGTGTATGGGTATTTTGCCTGCATATATGTCTATGCACCAT-
GTGTGTGCAGTGCTTGAG   FFFFFECGFDCFGDGDFEE@EEGIBFGGCGFFGACGFCGFFDGDGFFFFFFEGCDFCGFFGG@FFF=EFF-
DGGGGGFDGFFFGGGFGFFGGGFFGGGDFG    NH:i:1  HI:i:1  AS:i:88nM:i:0  Cx:i:7767       Cy:i:18052
UR:Z:7AE49      XF:i:0  GE:Z:Xkr4       GS:Z:-  UB:Z:79E49
E100026571L1C008R03103646063    16      1       3090385 255     16M224429N32M   *       0
0       AATGGATTATGTTGATGGTTGAATATATGGAAATCCATCAACATAATC        GGGGGGGGGGGGGGGGGGGGGGG-
FGGGGGGGGGGGGGGGGGGGGGGG       NH:i:1  HI:i:1  AS:i:36 nM:i:0  Cx:i:9239       Cy:i:8595
UR:Z:E1454      XF:i:1  GE:Z:Xkr4       GS:Z:-
```

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/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 { ( 73956787 ) / ( 73956787 ) }
   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 { ( 24606 ) / ( 24606 ) }
   DATA {
   (0): {
         "Gm1992",
         0,
         132
      },
   (1): {
```

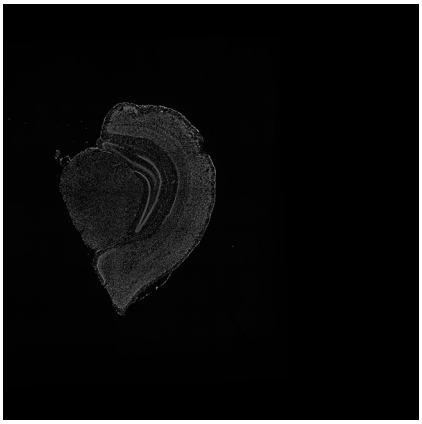
3.3.4 Example of count Sampling File

```
$ head -8 /path/to/output/03.count/SS200000135TL_D1_raw_barcode_gene_exp.txt
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
11152 6947 10551 322978 1
```

3.4 register

3.4.1 Registered image

File **/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
```

3.5 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): {
         6148,
         10906,
         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): {
         “0610005C13R
ik\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000”,
         0,
         25
      },
   (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

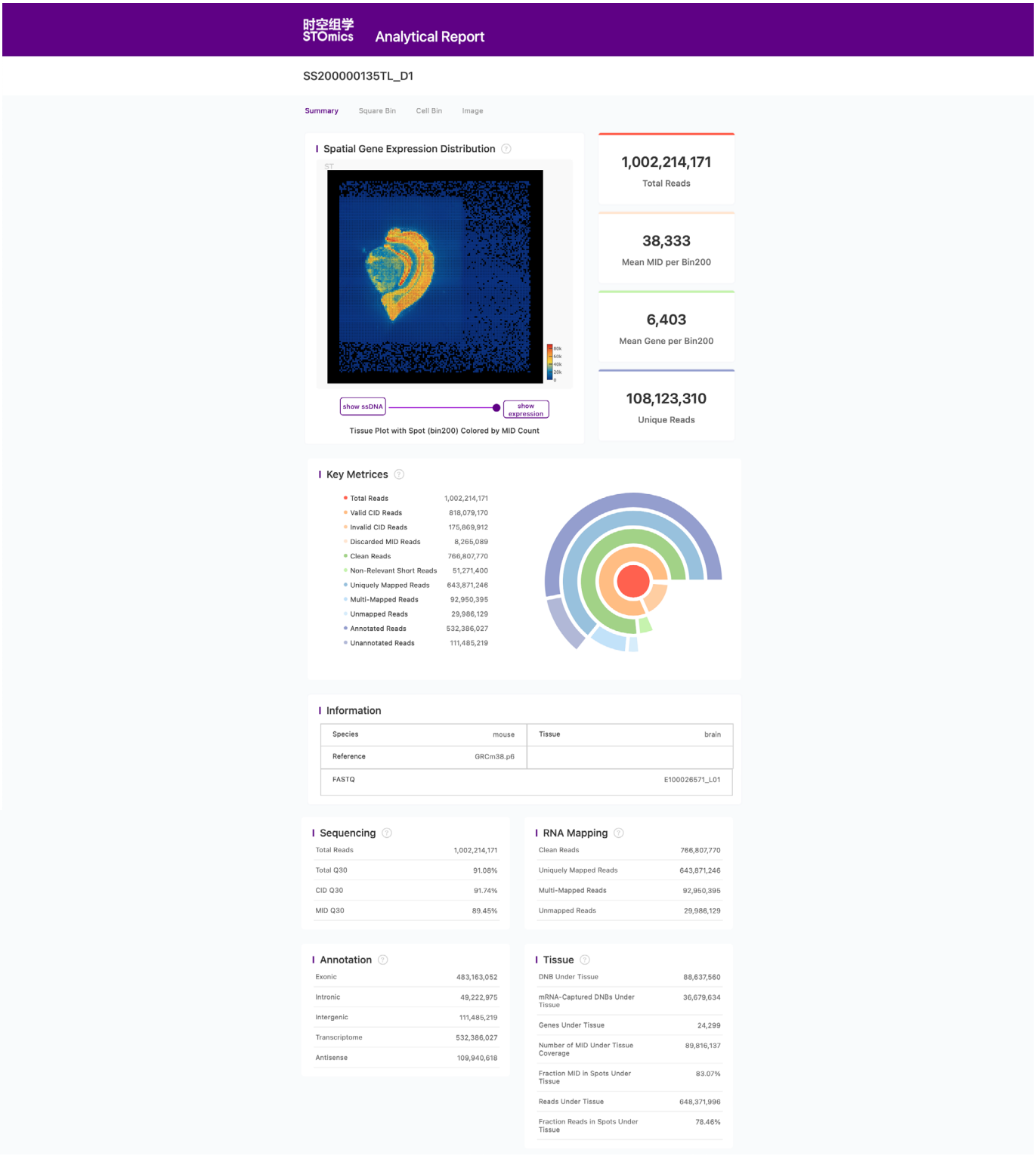
```
$ cat /path/to/output/07.saturation/sequence_saturation.tsv
sample  bar_x   bar_y1  bar_y2  bar_umi bin_x   bin_y1  bin_y2  bin_umi
0.05    26619302        0.249906        1       19966974        26619302
0.274024        3100    7309
0.1     53238604        0.389633        1       32495084        53238604
0.410009        4135    11896
0.2     106477208       0.542708        1       48691159        106477208
0.557736        5030    17825
0.3     159715808       0.625397        1       59830026        159715808
0.637281        5497    21903
0.4     212954416       0.677911        1       68590378        212954416
0.687768        5805    25110
0.5     266193008       0.71435 1       76038055        266193008
0.722835        6032    27836
0.6     319431616       0.741211        1       82665449        319431616
0.748708        6213    30262
0.7     372670208       0.762077        1       88666708        372670208
0.768813        6337    32459
0.8     425908832       0.77864 1       94279199        425908832
0.784793        6458    34514
0.9     479147392       0.792299        1       99519557        479147392
0.797978        6580    36433
1       532386027       0.803707        1       104503499       532386027
0.80899 6655    37933
```

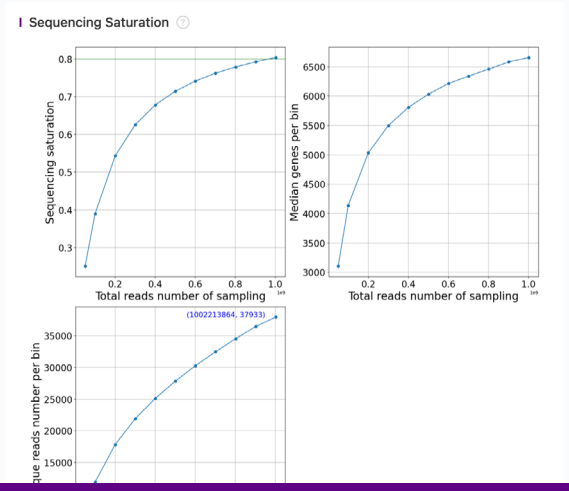
3.8 report

3.8.1 Example of Statistical Summary Report

```
$ head /path/to/output/08.report/SS200000135TL_D1.statistics.json
{
    “version”: “version_v2”,
    “1.Filter_and_Map”: {
        “1.1.Adapter_Filter”: [
            {
                “Sample_id”: “E100026571_L01”,
                “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

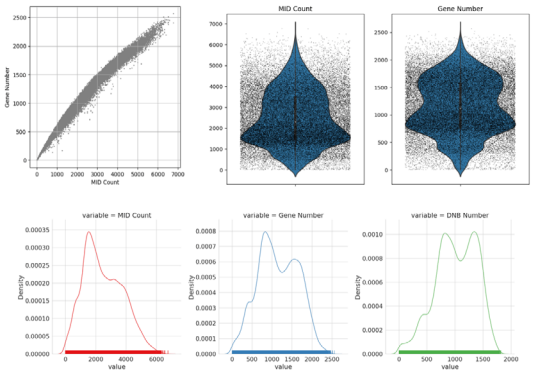
SS200000135TL_D1

Summary Square Bin Cell Bin Image

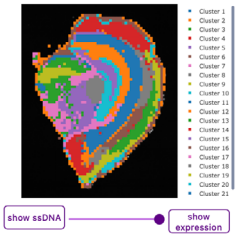
Tissue Square Bin Statistics

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

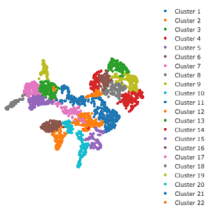
Bin 50



Clustering



Tissue Plot with Spots (bin200)



UMAP Projection of Spots (bin200)

时空组学 STOmics Analytical Report

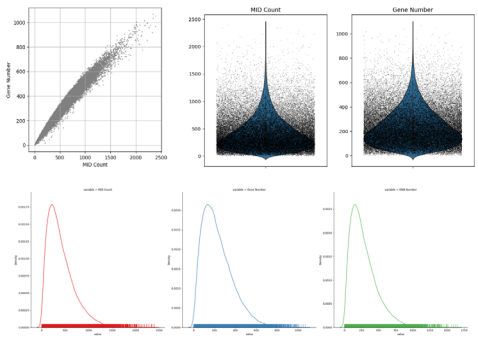
SS200000135TL_D1

Summary Square Bin Cell Bin Image

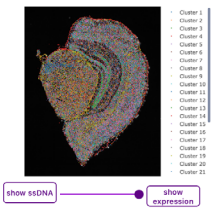
Cell Bin Statistics

Mean Cell Area	286	Median Cell Area	270
Mean Cell Boundary DNB Count	282	Median Cell Boundary DNB Count	238
Mean Gene Type	229	Median Gene Type	201
Mean MID	411	Median MID	343

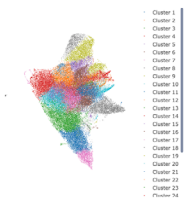
Cell Bin



Clustering



Tissue Plot with Spots (cell bin)



UMAP Projection of Spots (cellbin)

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	95
Registration Score	99

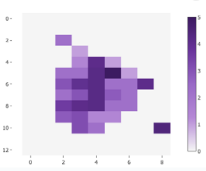
Stitching

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

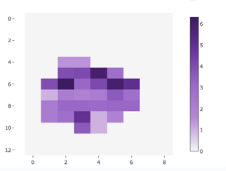
Registration

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: Recommend Directory Structure for Raw Data

```
$ 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
```

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

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.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 performing sequence saturation. The 5 columns are coorY, coorX, geneIndex, MIDIndex, readCount.
REGISTER & IPR2IMG	1.ini	Image configuration file.
	date-hh-mm-ss.log	Log file.
	3.jpg	Microscope stitched image.
	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.
	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.

STEP	OUTPUT	FILE DESCRIPTION
REGISTER & IPR2IMG	matrix_template.txt	Track line cross point template for the registered ssDNA image. Used for assessing registeration result.
	SN_date_time_version.csv	Clarity score for each small image.
	SN_date_time_version.ipr	IPR format image process record file which records basic image information gathered from imageQC.
	SN_date_time_version.png	Clarity score heatmap for each small image.
	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.
	SN_date_time_version.txt	Txt file records track line cross point coordinates.
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.
	dnb_merge/bin200.png	A thumbnail image converts from gene expression matrix in bin200.
	tissue_fig	This directory stores the statistical plots for the tissue coverage region
	SN.ssDNA.rpi	Registered panoramic image file (downsampled) that stores as an image pyramid. Same to the register output 7_result/SN.rpi
	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)

STEP	OUTPUT	FILE DESCRIPTION
TISSUECUT	violin_150x150_MID_gene.png	Violin plots show the distribution of deduplicated MID count and gene number in each bin (bin150)
	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.txt	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
	cellcut.stat	Statistical report for cell segmentation and expression.

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
...
```

bind file path before execute command.

```
$ export SINGULARITY_BIND="/path/to/file/directory"
```

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.

Appendix D: Manual Image Merge Using ImageJ

We provide step-by-step procedures demonstrating how to merge your SN_regist.tif image with your SN_tissue_cut.tif image or SN_mask.tif image to check for tissue segmentation and cell segmentation performance, respectively. Please make sure you have ImageJ installed on your computer.

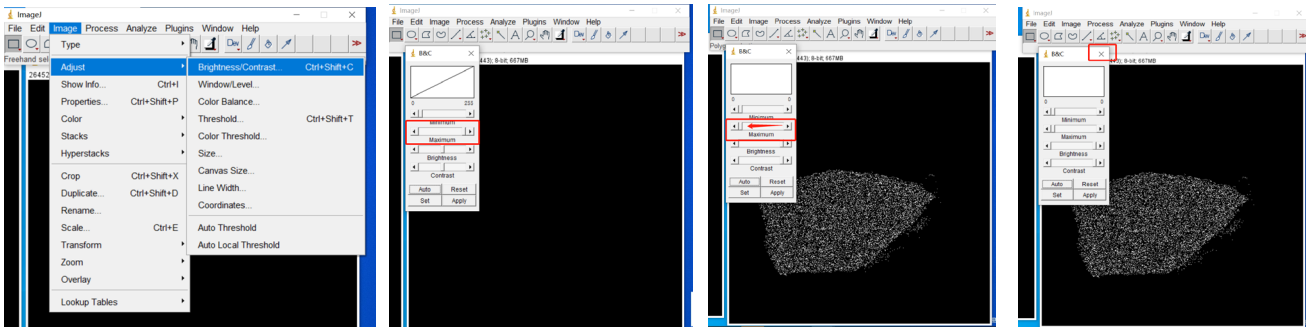
- Obtain your SN_regist.tif image and SN_mask.tif (SN_tissue_cut.tif) image from your output files generated after running **`ipr2img`**.

Here showing an example using

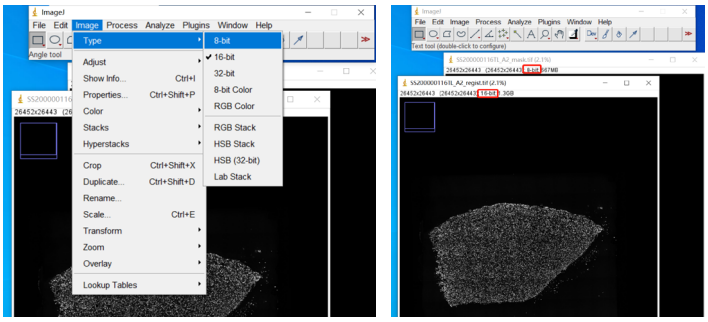
- **SS200000116TL_A2_regist.tif (post-registered ssDNA image, image bit depth of 16 bit)**
- **and SS200000116TL_A2_mask.tif (post-registered cellcut image mask, image bit depth of 8 bit).**

1) First , open both SN_regist.tif image and SN_mask.tif image using ImageJ.

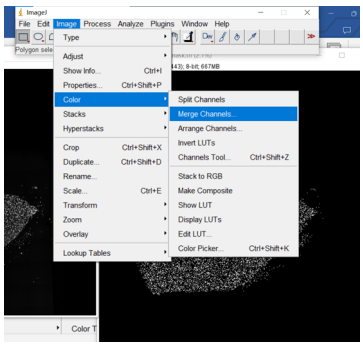
2) Once opened both files, if SN_mask.tif image is not showing, click **Image > Adjust > Brightness/Contrast**, and pull the maximum bar to the left. The image should appear. Do not click apply, just simply close the B&C interface.



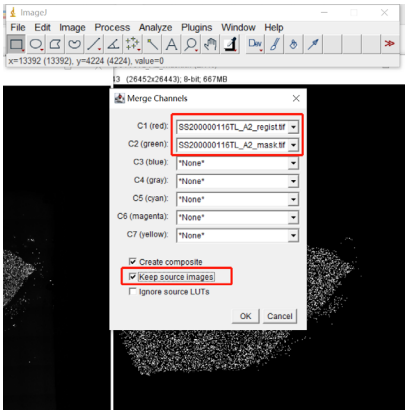
3) Now, if your SN_regist.tif and SN_mask.tif were set to different sizes, make sure to adjust the image that was in 16 bit to 8 bit. Click **Image > Type > 8 bit**.



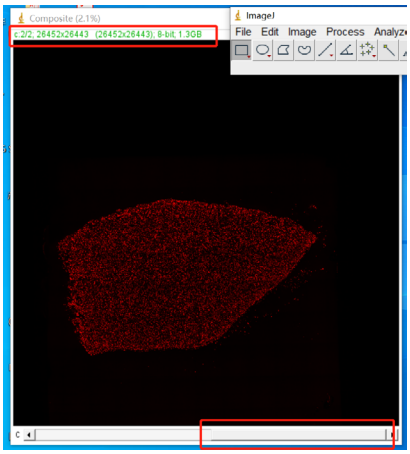
4) Merge your SN_regist.tif and SN_mask.tif images by selecting **Image > Color > Merge Channels**



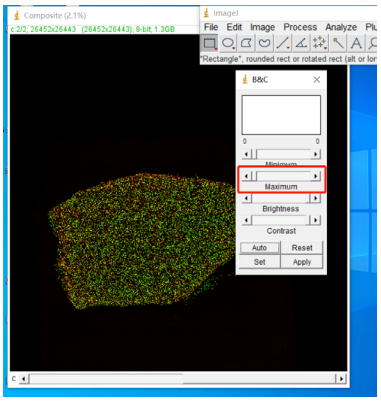
5) Assign your SN_regist.tif to the red channel, and your SN_mask.tif (or SN_tissue_cut.tif) to the green channel. Click the “Keep source images” box so your original images won’t be closed after the merged image is generated.



6) If green channel did not shown on your merged image, set the bottom bar to the green channel (image info shown in green once switched to green channel).

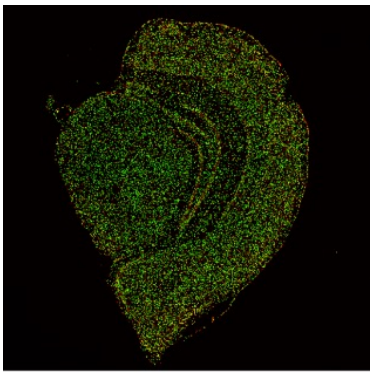
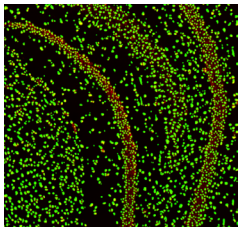
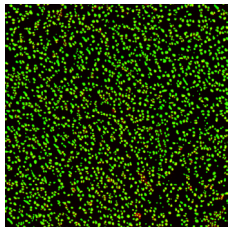
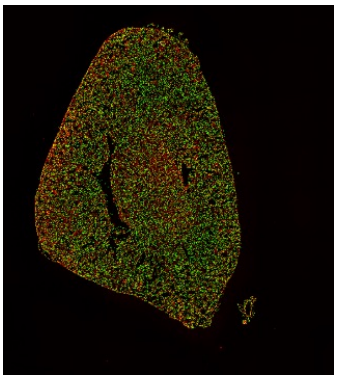
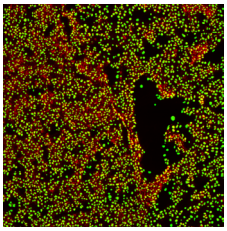
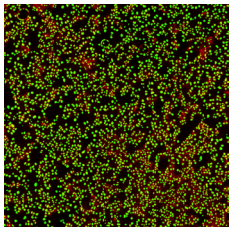


7) Go to **Image > Adjust > Brightness/Contrast**, pull the maximum bar to the left. Mask image of cell segmentation should appear in green.



8) Zoom in on your merged image and check if green area has more coverage than the red.

Examples of good cell segmentation vs bad cell segmentation:

Good	Bad
  	  
Overall segmentation of this sample is good as there is no large area coverage of red in the image. Besides the denser area within the tissue or tissue edge with strong exposure showing incomplete segmentation, the remaining area have clear boundaries of segmented cells, which could be determined as having good cell segmentation.	This is a kidney sample which exhibit very dense tissue morphology, resulting in poor overall segmentation performance and was determined as not suitable for further cell binning analysis

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. 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