时空组学
STOmics

STOMICS ANALYSIS WORKFLOW SOFTWARE SUITE

USER MANUAL

Software Version: V4.1.0

Manual Version: A2

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 improvements for `mapping`; updated gene annotation approach in `count`; updated image stitching and tissue segmentation model for better performance on image stitching and segmentation for tissue with voids.

Note: Please download the latest version of the manual and use it with the corresponding software version.

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

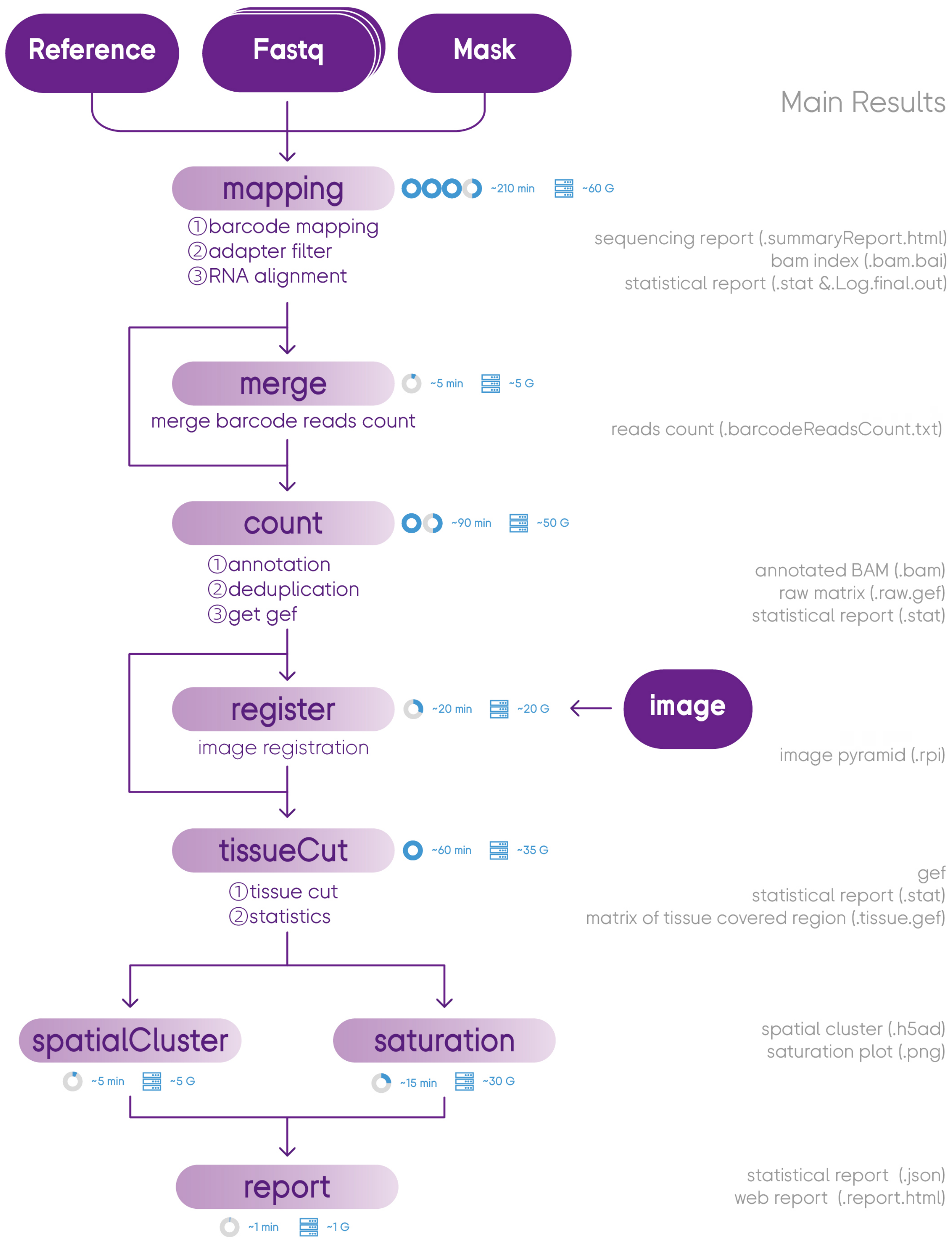


TABLE OF CONTENTS

CHAPTER 1: STOMICS ANALYSIS WORKFLOW SOFTWARE SUITE

1.1. Software Introduction 1
1.2. System Requirements 1
1.3. SAW Docker Image 2
1.4. SAW GitHub 3
1.5. SAW Test Data 3
1.6. SAW Output File Format 3

CHAPTER 2: SAW PIPELINES & ARGUMENTS

2.1. mapping 5
2.2. merge (Optional) 8
2.3. count 9
2.4. register (Optional) 10
2.5. tissueCut 12
2.6. spatialCluster 13
2.7. saturation 14
2.8. report 15
2.9. Other Applications 17

CHAPTER 3: TEST DATA DEMONSTRATION

3.1. mapping 21
3.2. merge 23
3.3. count 23
3.4. register 25
3.5. tissueCut 25
3.6. saturation 28
3.7. report 28

REFERENCES 31

CONTACT US 32

CHAPTER 1
STOMICS ANALYSIS WORKFLOW SOFTWARE SUITE

1.1. Software Introduction

STOmics 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 eight essential pipelines:

- **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 (optional):** Align microscopic tissue staining image with gene expression matrix file (GEF) generated from `count.`
- **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.
- **saturation:** Calculate sequencing saturation of tissue coverage area based on the file that used for sampling data generated from `count`.
- **spatialCluster:** Perform clustering analysis for spots (bin200) according to the gene expression matrix of the tissue covered area generated from `tissueCut`.
- **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, and clustering analysis results.

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 software:

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

1.3. SAW Docker Image Installation

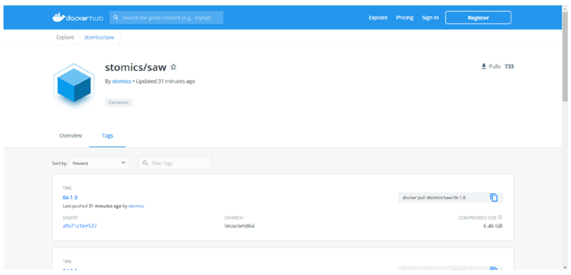
SAW is delivered as a docker image that bundles all of its required software dependencies. You can pull the SAW docker image from Docker Hub to your local system and run analyses offline.

Please download the latest version of the software and use it with the corresponding **STOmics Analysis Workflow Softwate 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 4.1.0 as an example.



stomics/saw
By stomics • Updated 31 minutes ago
Container
Pulls 733

TAG
04.1.0
Last pushed 31 minutes ago by stomics
docker pull stomics/saw:04.1.0

DIGEST	OS/ARCH	COMPRESSED SIZE
afb71a1ee537	linux/amd64	6.46 GB

TAG

1.3.1 SAW Installation via Singularity

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

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

```
$ singularity build SAW_v4.1.0.sif docker://stomics/saw:04.1.0  ## option 1
$ singularity build --sandbox SAW_v4.1.0/ docker://stomics/saw:04.1.0  ## option 2
```

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

```
$ export SINGULARITY_CACHEDIR=/path/to/build

$ singularity build --sandbox SAW_v4.1.0/ docker://stomics/saw:04.1.0
```

Step 2: Run pipelines (3 options):

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

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

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

```
$ /path/to/SAW_v4.1.0.sif <application>  ## option 1.1
$ singularity exec /path/to/SAW_v4.1.0.sif <application>  ## option 1.2
```

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

```
$ /path/to/SAW_v4.1.0.sif /bin/bash  ## option 2.1
Singularity>
Singularity> <shell-command>
Singularity> exit
$
$ singularity shell /path/to/SAW_v4.1.0.sif  ## option 2.2
Singularity>
Singularity> <shell-command>
Singularity> exit
$
$ singularity shell SAW_v4.1.0  ## option 2.3 for sandbox
Singularity>
Singularity> <shell-command>
Singularity> exit
$
```

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

```
*$ singularity shell -B /path/to/directory/on/the/host-machine:/path/to/directory/
 mounted/in/the/container/path/to/SAW_v4.1.0.sif
Singularity>
Singularity> <shell-command>
Singularity> exit
$
```

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

1.3.2 SAW Installation via Docker

Step 1: Pull the SAW docker image

```
$ docker pull stomics/saw:04.1.0
```

Step 2: Run pipelines interactively

```
$ docker run -it stomics/saw:04.1.0
```

1.4 SAW Github

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

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

Note! Please build your reference before running SAW analysis.

1.5 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

1.6 SAW Output File Format

Please check STOmics File Format Manual to get more information on SAW output files format.

CHAPTER 2
SAW PIPELINES & ARGUMENTS

2.1 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. Reads that get the coordinate annotations are Valid CID mRNA Reads (**Valid CID Reads**). After filtering, 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 Mask file **(.h5)**
- Indexed reference genome
- bcPara file **(.bcPara)**, please check the content of Table 2-2

Expected running time for ~1G reads: ~3.5 h, Memory: ~60G

NOTE! Before proceeding, if users need to estimate the number of CIDs and the memory needed for running `mapping`, please refer to 2.9.1 CIDcount for more information.

2.1.1 Arguments and Options

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

Table 2-1 `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.
`--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 2-2 `mapping --stParaFile` Arguments and Options

Parameter	Function
`in`	(Required) Path to the STOmics Chip Mask file.
`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. 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.
`platform`	(Optional) Sequencing platform. Valid options: SEQ500, T1, or T10.
`barcodeStart`	(Required; defaults to 0) CID start position.
`barcodeLen`	(Required; defaults to 25) CID length.
`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.
`bcNum`	(Optional) CID count. Please check 2.9.1 CIDcount for more information.

2.1.2 Usage Example

Prepare **`mapping`** --stParaFile input file `{lane}.bcPara` as shown below:

Note! Replace `{SN}` with your STOmics Chip 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.

```
$ mkdir /path/to/output/00.mapping
$ vim /path/to/output/00.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/00.mapping/{lane}.barcodeReadsCount.txt
action=4
platform=T10
barcodeStart=0
barcodeLen=25
umiStart=25
umiLen=10
umiRead=1
mismatch=1
```

Run **mapping** pipeline

```
$ singularity exec SAW_v4.1.0.sif mapping \
                --outSAMattributes spatial \
                --outSAMtype BAM SortedByCoordinate \
                --genomeDir /path/to/genomeDir \
                --runThreadN 8 \
                --outFileNamePrefix /path/to/output/00.mapping/{lane}. \
                --sysShell /bin/bash \
                --stParaFile /path/to/output/00.mapping/{lane}.bcPara \
                --readNameSeparator \" \" \
                --limitBAMsortRAM 38582880124 \
                --limitOutSJcollapsed 10000000 \
                --limitIObufferSize=280000000 \
                --outBAMsortingBinsN 50 \
                > /path/to/output/00.mapping/{lane}_barcodeMap.stat
```

Note! If a fatal error has occurred as shown below, you could delete --readNameSeparator \" \" from the command:

```
EXITING: FATAL INPUT ERROR: empty value for parameter "readNameSeparator" in input
"Command-Line"

SOLUTION: use non-empty value for this parameter
```

2.1.3 Outputs

The output files of **mapping** are organized as below:

```
$ ll -Rth /path/to/output/00.mapping
-rw-rw-r--  1 ubuntu ubuntu  11M Apr 13 20:57 lane.Aligned.sortedByCoord.out.bam.bai
-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
```

If one sample has multiple FASTQ files, you need to run **mapping** for each FASTQ pair. The output files are organized as below (Next page showing example of 2 pairs of FASTQ):

```
$ tree /path/to/multi_lane_output/00.mapping
/path/to/multi_lane_output/00.mapping
└── 00.mapping
    ├── lane1.Aligned.sortedByCoord.out.bam
    ├── lane1.Aligned.sortedByCoord.out.bam.bai
    ├── 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.bai
    ├── lane2_barcodeMap.stat
    ├── lane2.barcodeReadsCount.txt
    ├── lane2.bcPara
    ├── lane2.Log.final.out
    ├── lane2.Log.out
    ├── lane2.Log.progress.out
    └── lane2.SJ.out.tab
```

2.2 merge (optional)

SAW **`merge`** pipeline is used to combine the results of **`mapping`**.
Run **`merge`** requires the following files:

- **`mapping`** output mapped CID with reads count files **(.txt)**

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

2.2.1 Arguments and Options

Table 2-3 `merge` Arguments and Options

Parameter	Function
`-i`	(Required) Mapped CID list files with reads counts for each CID.
`--out`	(Required) Mapped CID list file which merges all input files.
`--action`	(Required; defaults to 1) Action number. Valid options: 1=map CID to slide, 2=merge CID list, 3=mask format change, or 4=mask merge.

2.2.2 Usage Example

```
$ mkdir /path/to/multi_lane_output/01.merge
$ singularity exec SAW_v4.1.0.sif merge \
                -i /path/to/multi_lane_output/00.mapping/{lane1}.barcodeReadsCount.
txt,/path/to/multi_lane_output/00.mapping/{lane2}.barcodeReadsCount.txt \
                --out /path/to/multi_lane_output/01.merge/{SN}.barcodeReadsCount.txt
                --action 2
```

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

2.2.3 Ouputs

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

```
$ ll -Rth /path/to/multi_lane_output/01.merge
-rw-rw-r--  1 ubuntu ubuntu 997M Apr 13 20:53 SN.barcodeReadsCount.txt
```

2.3 count

SAW **count** pipeline annotates **Uniquely Mapped Reads** filtered from **mapping** output based on the reference genome annotation records. Through quantification of annotated reads, **count** generates spatial gene expression data after de-duplication reads according to CID, gene ID and MID information.

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: 1.5h, Memory: ~50G

2.3.1 Arguments and Options

Table 2-4 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 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) STOmics Chip serial number (SN).
-c	(Optional) CPU core number to use.
--save_lq	(Optional) Save low quality reads if set.
--save_dup	(Optional) Save duplicate reads if set.
--umi_on	(Optional) Correct MID if set.
--sat_file	(Optional) Set the saturation sampling file name which is prepared for sequencing saturation (requires --umi_on).
-m	(Optional; defaults to detected) Set available memory (GB).

2.3.2 Usage Example

```
$ mkdir -p /path/to/output/02.count
$ geneExp=/path/to/output/02.count/{SN}.raw.gef
$ saturationSamplingFile=/path/to/output/02.count/raw_barcode_gene_exp.txt
$ singularity exec SAW_v4.1.0.sif count \
        -i /path/to/output/00.mapping/{lane}.Aligned.sortedByCoord.out.bam \
        -o /path/to/output/02.count/{SN}.Aligned.sortedByCoord.out.merge.q10.dedup.
target.bam \
        -a /path/to/genomeDir/genes.gtf \
        -s /path/to/output/02.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/02.count
$ geneExp=/path/to/multi_lane_output/02.count/{SN}.raw.gef
$ saturationSamplingFile=/path/to/multi_lane_output/02.count/raw_barcode_gene_exp.txt
$ singularity exec SAW_v4.1.0.sif count \
        -i /path/to/multi_lane_output/00.mapping/{lane1}.Aligned.sortedByCoord.out.
bam,/path/to/multi_lane_output/00.mapping/{lane2}.Aligned.sortedByCoord.out.bam \
        -o /path/to/multi_lane_output/02.count/{SN}.Aligned.sortedByCoord.out.merge.
q10.dedup.target.bam \
        -a /path/to/genomeDir/genes.gtf \
        -s /path/to/multi_lane_output/02.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.

2.3.3 Ouputs

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

```
$ ll -Rth /path/to/output/02.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
```

2.4 register (optional)

SAW **register** pipeline aligns the microscopic tissue staining image with the plot of the gene expression matrix generate by **count** based on the tracklines on the chip while establishing the mapping relationship between images and spatial expression.

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

2.4.1 Arguments and Options

Table 2-5 `register` Arguments and Options

Parameter	Function
-i	(Required) ImageQC processed staining image TAR.GZ file.
-c	(Required) ImageQC JSON file.
-v	(Required) `count` output gene expression matrix GEF file.
-o	(Required) Path to the directory where to store the `register` outputs.

2.4.2 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}*.json | head -1)
$ mkdir -p /path/to/output/03.register
$ singularity exec SAW_v4.1.0.sif register \
                  -i ${image4register} \
                  -c ${imageQC} \
                  -v /path/to/output/02.count/{SN}.raw.gef \
                  -o /path/to/output/03.register
```

2.4.3 Outputs

`register` output files are organized as below:

```
$ ll -th /path/to/output/03.register
drwxrwxr-x  2 ubuntu ubuntu 4.0K Apr 13 22:15 7_result/
drwxrwxr-x  2 ubuntu ubuntu 4.0K Apr 13 22:15 6_analysis/
drwxrwxr-x  2 ubuntu ubuntu 4.0K Apr 13 22:15 3_vision/
drwxrwxr-x  2 ubuntu ubuntu 4.0K Apr 13 22:15 4_register/
drwxrwxr-x  2 ubuntu ubuntu 4.0K Apr 13 22:13 5_mask/
drwxrwxr-x  2 ubuntu ubuntu 4.0K Apr 13 22:12 2_stitch/
drwxrwxrwx  3 ubuntu ubuntu 4.0K Apr 13 22:11 1_origin/
...
$ ll -Rth /path/to/output/03.register/4_register
-rw-rw-r-- 1 ubuntu ubuntu 668M Apr 13 22:15 fov_stitched_regist.tif
-rw-rw-r-- 2 ubuntu ubuntu 2.1M Apr 13 22:14 transform_thumb.png
-rw-rw-r-- 2 ubuntu ubuntu   39 Apr 13 22:14 attrs.json
-rw-rw-r-- 1 ubuntu ubuntu 458M Apr 13 22:13 fov_stitched_transformed.tif
-rw-rw-r-- 1 ubuntu ubuntu   50 Apr 13 22:13 im_shape.txt
$
$ ll -Rth /path/to/output/03.register/5_mask
-rw-rw-r-- 1 ubuntu ubuntu 458M Apr 13 22:13 fov_stitched_transformed_tissue_cut.tif
...
$ ll -Rth /path/to/output/03.register/7_result
...
-rw-rw-r-- 1 ubuntu ubuntu 321M Apr 13 22:15 merge_SN.tif
-rw-rw-r-- 1 ubuntu ubuntu 176M Apr 13 22:15 SN.rpi
-rw-rw-r-- 1 ubuntu ubuntu 4.9K Apr 13 22:15 SN_20211125.json
-rw-rw-r-- 1 ubuntu ubuntu   38 Apr 13 22:15 SN_tissue_bbox.csv
-rw-rw-r-- 1 ubuntu ubuntu 668M Apr 13 22:15 SN_tissue_cut.tif
-rw-rw-r-- 1 ubuntu ubuntu 668M Apr 13 22:15 SN_regist.tif
```

2.5 tissueCut

SAW **tissueCut** pipeline could delineate and extract the tissue coverage area based on the aligned image generated from **register** 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 with reads count file **(.txt)**
- **count** output gene expression matrix file **(.raw.gef)**
- Directory stores aligned microscopic staining image **(optional)**

Expected running time for ~1G reads: ~1h, Memory: ~35G

2.5.1 Arguments and Options

Table 2-6 tissueCut Arguments and Options

Parameter	Function
--dnbfile	(Required) Mapped CID list file with reads counts for each CID. Use the merged mapped CID with reads count file if multiple pairs of FASTQ files are involved in the analysis.
-i	(Required) count output gene expression matrix GEF 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) STOmics Chip serial number (SN).
--platform	(Required, default to T1) Sequencing platform.

2.5.2 Usage Example

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

```
$ mkdir -p /path/to/output/04.tissuecut
$ singularity exec SAW_v4.1.0.sif tissueCut \
        --dnbfile /path/to/output/01.merge/{SN}.barcodeReadsCount.txt \
        -i /path/to/output/02.count/{SN}.raw.gef \
        -o /path/to/output/04.tissuecut \
        -s /path/to/output/03.register/7_result \
        -t tissue \
        --snId {SN} \
        --platform T10
```

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

```
$ mkdir -p /path/to/output/04.tissuecut
$ singularity exec SAW_v4.1.0.sif tissueCut \
        --dnbfile /path/to/output/01.merge/{SN}.barcodeReadsCount.txt \
        -i /path/to/output/02.count/{SN}.raw.gef \
        -o /path/to/output/04.tissuecut \
        -t tissue \
        --snId {SN} \
        --platform T10
```

2.5.3 Outputs

`tissueCut` output files:

Image is provided:

```
$ ll -th /path/to/output/04.tissuecut
-rw-rw-r--  1 ubuntu ubuntu 1.3K Apr 13 22:39 tissuecut.stat
drwxrwxr-x  2 ubuntu ubuntu 4.0K Apr 13 22:39 tissue_fig/
-rw-rw-r--  1 ubuntu ubuntu 526M Apr 13 22:19 SN.tissue.gef
drwxrwxr-x  2 ubuntu ubuntu 4.0K Apr 13 22:17 dnb_merge/
-rw-rw-r--  2 ubuntu ubuntu 5.6G Apr 13 22:17 SN.gef
$
$ ll -Rth /path/to/output/04.tissuecut/tissue_fig
-rw-rw-r-- 1 ubuntu ubuntu 136K Apr 13 22:39 violin_200x200_MID_gene.png
-rw-rw-r-- 1 ubuntu ubuntu  27K Apr 13 22:39 scatter_200x200_MID_gene_counts.png
-rw-rw-r-- 1 ubuntu ubuntu 185K Apr 13 22:36 violin_150x150_MID_gene.png
-rw-rw-r-- 1 ubuntu ubuntu  32K Apr 13 22:36 scatter_150x150_MID_gene_counts.png
-rw-rw-r-- 1 ubuntu ubuntu 295K Apr 13 22:32 violin_100x100_MID_gene.png
-rw-rw-r-- 1 ubuntu ubuntu  29K Apr 13 22:32 scatter_100x100_MID_gene_counts.png
-rw-rw-r-- 1 ubuntu ubuntu 514K Apr 13 22:28 violin_50x50_MID_gene.png
-rw-rw-r-- 1 ubuntu ubuntu  29K Apr 13 22:28 scatter_50x50_MID_gene_counts.png
-rw-rw-r-- 2 ubuntu ubuntu 176M Apr 13 22:17 SN.ssDNA.rpi
```

Image is not provided:

```
$ tree /path/to/output/04.tissuecut
/path/to/output/04.tissuecut
├── dnb_merge
|   ├── bin200.png
├── SN.gef
├── SN.tissue.gef
├── tissue_fig
|   ├── scatter_100x100_MID_gene_counts.png
|   ├── scatter_100x100_MID_gene_counts.png
|   ├── scatter_150x150_MID_gene_counts.png
|   ├── scatter_200x200_MID_gene_counts.png
|   ├── scatter_50x50_MID_gene_counts.png
|   ├── violin_100x100_MID_gene.png
|   ├── violin_150x150_MID_gene.png
|   ├── violin_200x200_MID_gene.png
|   └── violin_50x50_MID_gene.png
└── tissuecut.stat
```

2.6 spatialCluster

SAW `spatialCluster` pipeline perform clustering analysis for spots at bin200 (bin size of 200) using 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: ~5 min, Memory: ~5G

2.6.1 Arguments and Options

Table 2-7 `spatialCluster` Arguments and Options

Parameter	Function
-i	(Required) tissuCut output GEF file for the tissue coverage area.
-o	(Required) Output path for the clustering result in H5AD format.
-s	(Required) Bin size, recommend to use 200.

2.6.2 Usage Example

```
$ mkdir -p /path/to/output/05.spatialcluster
$ singularity exec SAW_v4.1.0.sif spatialCluster \
    -i /path/to/output/04.tissuecut/{SN}.tissue.gef \
    -o /path/to/output/05.spatialcluster/{SN}.spatial.cluster.h5ad \
    -s 200
```

2.6.3 Outputs

`spatialCluster` output files are:

```
$ ll -Rht /path/to/output/05.spatialcluster
-rw-rw-r--  2 ubuntu ubuntu 231M Apr 13 22:40 SN.spatial.cluster.h5ad
```

2.7 Saturation

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

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

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

Expected running time for ~1G reads: ~15 min, Memory: ~30G

2.7.1 Arguments and Options

Table 2-8 `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.
--bcstat	(Required) `mapping` output statistical report of CID mapping. Separate multiple files by comma.
--summary	(Required) `count` output statistical report of annotation.

2.7.2 Usage Example

```
$ mkdir -p /path/to/output/06.saturation
$ singularity exec SAW_v4.1.0.sif saturation \
      -i /path/to/output/02.count/raw_barcode_gene_exp.txt \
      --tissue /path/to/output/04.tissuecut/{SN}.tissue.gef \
      -o /path/to/output/06.saturation \
      --bcstat /path/to/output/00.mapping/{lane}_barcodeMap.stat \
      --summary /path/to/output/02.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/06.saturation
$ singularity exec SAW_v4.1.0.sif saturation \
      -i /path/to/multi_lane_output/02.count/raw_barcode_gene_exp.txt \
      --tissue /path/to/multi_lane_output/04.tissuecut/{SN}.tissue.gef \
      -o /path/to/multi_lane_output/06.saturation \
      --bcstat /path/to/multi_lane_output/00.mapping/{lane1}_barcodeMap.stat,/path/
* to/multi_lane_output/00.mapping/{lane2}_barcodeMap.stat \
      --summary /path/to/multi_lane_output/02.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.

2.7.3 Outputs

`saturation` output files are:

```
$$ ll -Rht /path/to/output/06.saturation
-rw-rw-r--  1 ubuntu ubuntu  77K Apr 13 22:54 plot_200x200_saturation.png
-rw-rw-r--  1 ubuntu ubuntu  36K Apr 13 22:54 plot_1x1_saturation.png
-rw-rw-r--  1 ubuntu ubuntu  780 Apr 13 22:53 sequence_saturation.txt
```

2.8 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, and clustering analysis result plots.

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

- **`mapping`** output statistical reports of CID mapping **(.stat)** and **STAR** alignment **(.out)**
- **`count`** output statistical report of annotation **(.stat)**
- **`tissueCut`** output GEF file **(.tissue.gef)**, statistical report of tissue-covered region **(.stat)**, plots **(.png)** and image pyramid RPI file **(.rpi)**
- **`spatialCluster`** output clustering file **(.h5ad)**
- **`saturation`** output bin200 sequence saturation plot **(.png)**

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

2.8.1 Arguments and Options

Table 2-9 `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) `tissueCut` output GEF file.
`-b`	(Required) `tissueCut` output bin 200 scatter plot.
`-v`	(Required) `tissueCut` output bin 200 violin plots.
`-i`	(Optional) The image pyramid RPI file.
`-d`	(Required) `spatialCluster` output H5AD file.
`-t`	(Required) `saturation` output bin 200 sequence saturation plot.
`-r standard_version`	(Required) Set to specifying report version.
`-s`	(Required) The STOmics Chip serial number.
`--pipelineVersion`	(Required) Set to specifying analysis pipeline version.
`-o`	(Required) The directory to store outputs.

2.8.2 Usage Example

Run **`report`** if **`register`** aligned microscopic staining image is provided:

```
$ mkdir -p /path/to/output/07.report
$ singularity exec SAW_v4.1.0.sif report \
    -m /path/to/output/00.mapping/{lane}_barcodeMap.stat \
    -a /path/to/output/00.mapping/{lane}.Log.final.out \
    -g /path/to/output/02.count/{SN}.Aligned.sortedByCoord.out.merge.q10.dedup.
target.bam.summary.stat \
    -l /path/to/output/04.tissuecut/tissuecut.stat \
    -n /path/to/output/04.tissuecut/{SN}.gef \
    -d /path/to/output/05.spatialcluster/{SN}.spatial.cluster.h5ad \
    -t /path/to/output/06.saturation/plot_200x200_saturation.png \
    -b /path/to/output/04.tissuecut/tissue_fig/scatter_200x200_MID_gene_counts.png \
    -v /path/to/output/04.tissuecut/tissue_fig/violin_200x200_MID_gene.png \
    -r standard_version \
    -i /path/to/output/04.tissuecut/tissue_fig/{SN}.ssDNA.rpi \
    -s {SN} \
    --pipelineVersion SAW_v4.1.0 \
    -o /path/to/output/07.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/07.report
$ singularity exec SAW_v4.1.0.sif report \
    -m /path/to/multi_lane_output/00.mapping/{lane1}_barcodeMap.stat,/path/to/multi_
lane_output/00.mapping/{lane2}_barcodeMap.stat \
    -a /path/to/multi_lane_output/00.mapping/{lane1}.Log.final.out,/path/to/multi_
lane_output/00.mapping/{lane2}.Log.final.out \
    -g /path/to/multi_lane_output/02.count/{SN}.Aligned.sortedByCoord.out.merge.q10.
dedup.target.bam.summary.stat \
    -l /path/to/multi_lane_output/04.tissuecut/tissuecut.stat \
    -n /path/to/multi_lane_output/04.tissuecut/{SN}.gef \
    -d /path/to/multi_lane_output/05.spatialcluster/{SN}.spatial.cluster.h5ad \
    -t /path/to/multi_lane_output/06.saturation/plot_200x200_saturation.png \
    -b /path/to/multi_lane_output/04.tissuecut/tissue_fig/scatter_200x200_MID_gene_
counts.png \
    -v /path/to/multi_lane_output/04.tissuecut/tissue_fig/violin_200x200_MID_gene.png
\
    -r standard_version \
    -i /path/to/multi_lane_output/04.tissuecut/tissue_fig/{SN}.ssDNA.rpi \
    -s {SN} \
    --pipelineVersion SAW_v4.1.0 \
    -o /path/to/multi_lane_output/07.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 not provided (Here showing an example of just one pair of FASTQ, similar to multiple pairs),

```
$ mkdir -p /path/to/output/07.report
$ singularity exec SAW_v4.1.0.sif report \
    -m /path/to/output/00.mapping/{lane}_barcodeMap.stat \
    -a /path/to/output/00.mapping/{lane}.Log.final.out \
    -g /path/to/output/02.count/{SN}.Aligned.sortedByCoord.out.merge.q10.dedup.
target.bam.summary.stat \
    -l /path/to/output/04.tissuecut/tissuecut.stat \
    -n /path/to/output/04.tissuecut/{SN}.gef \
    -d /path/to/output/05.spatialcluster/{SN}.spatial.cluster.h5ad \
    -t /path/to/output/06.saturation/plot_200x200_saturation.png \
    -b /path/to/output/04.tissuecut/tissue_fig/scatter_200x200_MID_gene_counts.png \
    -v /path/to/output/04.tissuecut/tissue_fig/violin_200x200_MID_gene.png \
    -r standard_version \
    -s {SN} \
    --pipelineVersion SAW_v4.1.0 \
    -o /path/to/output/07.report
```

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

2.8.3 Outputs

`report` output files are organized as below:

```
$ ll -Rth /path/to/output/07.report
-rw-rw-r--  2 ubuntu ubuntu 1.2M Apr 13 22:54 SN.report.html
-rw-rw-r--  2 ubuntu ubuntu 5.8K Apr 13 22:54 new_final_result.json
```

2.9 Other Applications

2.9.1 CIDcount

`CIDcount` is a small program for computing the number of CIDs in the STOmics Chip Mask file and roughly estimating how much memory will be needed to do **`mapping`**.

```
$ singularity exec SAW_v4.1.0.sif CIDcount \
      -i /path/to/data/{SN}.barcodeToPos.h5 \  ## STOmics Chip Mask file path
      -s {speciesName} \  ## a string of 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}"
```

The output of **`CIDcount`**is shown as below,

```
$ singularity exec SAW_v4.1.0.sif CIDcount -i SN.barcodeToPos.h5 -s mouse -g 3
645784920  ## CID count
62  ## estimated memory for mapping
```

If users wich to run **`CIDcount`**, they may add "bcNum" to the required **`{lane}.bcPara`** file for **`mapping`** as shown below,

```
$ vim /path/to/output/00.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/00.mapping/{lane}.barcodeReadsCount.txt
action=4
platform=T10
barcodeStart=0
barcodeLen=25
umiStart=25
umiLen=10
umiRead=1
mismatch=1
bcNum=645784920  ## first line from output of CIDcount
```

2.9.2 gefTools

`gefTools`[12] is 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 its python encapsulated package **`gefpy`**[13].

Function 1: GEF to plain table GEM format

```
$ singularity exec SAW_v4.1.0.sif gefTools view \  ## convert GEF that only contains
bin1 geneExp
      -i /path/to/output/02.count/{SN}.raw.gef \
      -o {SN}.raw.gem
$ singularity exec SAW_v4.1.0.sif gefTools view \  ## convert a whole GEF
      -i /path/to/output/04.tissuecut/{SN}.gef \
      -o {SN}.gem
$ singularity exec SAW_v4.1.0.sif gefTools view \  ## convert tissue GEF that only
contains bin1 geneExp
      -i /path/to/output/04.tissuecut/{SN}.tissue.gef \
      -o {SN}.tissue.gem
```

Function 2: completion of a GEF

```
$ singularity exec SAW_v4.1.0.sif gefTools 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/02.count/{SN}.tissue.gef \
      -o {SN}.tissue.complete.gef \
      -b 1,20,50,100
```

Function 3: converting GEM to GEF

```
$ singularity exec SAW_v4.1.0.sif gefTools 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
```

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/04.tissue/{SN}.tissue.gef’ ,bin_size=200,n_thread=4)
>>> bgef.to_gem( ‘{SN}.tissue.bin200.gem’ )
```

CHAPTER 3
TEST DATA
DEMONSTRATION

Users may refer to this section as a format for testing SAW process. This chapter includes the statistical results and examples of critical files for each key step.

SN: SS200000135TL_D1

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

3.1 mapping

3.1.1 Statistical Report for CID Mapping and Filtering

```
$ cat /path/to/output/00.mapping/E100026571_L01_trim_read_barcodeMap.stat
...
getBarcodePositionMap_uniqBarcodeTypes: 645784920
total_reads:    1002214171
fixed_sequence_contianing_reads:        0       0.00%
pass_filter_reads:      1002214171
mapped_reads:   845170516       84.33%
reads_with_adapter:     8137401 0.81%
reads_with_dnb: 45284608        4.52%
barcode_exactlyOverlap_reads:   698287595       69.67%
barcode_misOverlap_reads:       146882921       14.66%
barcode_withN_reads:    0       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:       8451821 0.84%
umi_with_N_reads:       13355   0.00%
umi_with_polyA_reads:   13044   0.00%
umi_with_low_quality_base_reads:        8425422 0.84%
mapped_dnbs: 78023582
...
```

3.1.2 Statistical Report for Reference Genome Alignment

```
$ cat /path/to/output/00.mapping/E100026571_L01_trim_read.Log.final.out
...
                          Number of input reads |       783296686
                      Average input read length |       100
                                    UNIQUE READS:
                   Uniquely mapped reads number |       624859097
                        Uniquely mapped reads % |       79.77%
                          Average mapped length |       98.02
                       Number of splices: Total |       68091842
            Number of splices: Annotated (sjdb) |       65321302
                       Number of splices: GT/AG |       66183758
                       Number of splices: GC/AG |       462810
                       Number of splices: AT/AC |       43237
               Number of splices: Non-canonical |       1402037
                      Mismatch rate per base, % |       0.58%
                         Deletion rate per base |       0.07%
                        Deletion average length |       3.87
                        Insertion rate per base |       0.04%
                       Insertion average length |       1.26
                             MULTI-MAPPING READS:
        Number of reads mapped to multiple loci |       81769070
             % of reads mapped to multiple loci |       10.44%
        Number of reads mapped to too many loci |       2071180
             % of reads mapped to too many loci |       0.26%
                                  UNMAPPED READS:
  Number of reads unmapped: too many mismatches |       0
       % of reads unmapped: too many mismatches |       0.00%
            Number of reads unmapped: too short |       73977028
                 % of reads unmapped: too short |       9.44%
                Number of reads unmapped: other |       620311
                     % of reads unmapped: other |       0.08%
                                  CHIMERIC READS:
                       Number of chimeric reads |       0
                            % of chimeric reads |       0.00%
```

3.1.3 Example of BAM mapping

```
$ samtools view /path/to/output/00.mapping/E100026571_L01_trim_read.Aligned.
sortedByCoord.out.bam | head -3
E100026571L1C009R00301275185    16      1       3000095 255     26M121066N74M   *
0       0       GGCTTTTTTTTTTTTTTTTTTTTTTTTTTTCTAAATATTGGGTTTTATTAGCACCATGATAACT-
GTATATTAATTTGCACTGACTGTCATAACAAAATAC       G+:GFFGGFGFFGFFGFGGFFGFFFFFCFGFCFGGGF-
GGFGFFFFGGFGGFGFFFGGFFGFFFGFGFGFFGFFGFGFFFFGFFFFFFFFGGFFGGFFGEF    NH:i:1  HI:i:1
AS:i:88 nM:i:0  Cx:i:4826       Cy:i:11598      UR:Z:6FA29
E100026571L1C008R02501359039    256     1       3000101 3       11S33M329303N56M
*       0       0       CTCGACGGCCATTTTTTTTTTTTTTTTTTTTTTTGGGTGGTAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA       D=AEF?;F8FFDGGCEDDFBD>FB8FFFGG-
7G=5F2A9F2,3,F8EDFGGD@8GFGGFFAFFG@E?EFFCGF8GGFGFGFEA7G@FGECFFGGDF;FA<G    NH:i:2
HI:i:2  AS:i:66 nM:i:9  Cx:i:13900      Cy:i:20099      UR:Z:BA677
E100026571L1C007R00303973559    256     1       3000644 3       100M    *       0
0       GCCTCATTGTGCCCCATATGTTTGCCTATGTTGTGGACTTATTTTCATTAAACTTTAAAACATCTTTA-
ATTTTTTTCTTTATTTCATCATTGACCAAGCT       -FCA9D?GFFD<-D<CGFEGD-DG*FGFDFBE;E(9BGGE38FFF-
G9GG;0?GGFGB?E@G:GGG3GF79F0GGDG?G<D>F;EG,G?<<CD4>G=>B+C    NH:i:2  HI:i:2  AS:i:94
nM:i:2  Cx:i:8839       Cy:i:7539       UR:Z:120CF
```

3.2 merge

3.2.1 Example of Mapped CID List with Reads Count File

```
$ head /path/to/output/01.merge/SS200000135TL_D1.barcodeReadsCount.txt
12286   19289   1
2055    21686   3
10005   14086   2
5040    12492   1
15271   10095   6
6032    10419   1
7498    14163   1
15553   7772    2
3206    13520   1
13437   11123   3
```

3.3 count

3.3.1 Statistical Report for MID Filtering and Gene Annotation

```
$ cat /path/to/output/02.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
706628167       624859097       506676745       104709429       11.57   18.91   79.33
## ANNOTATION METRICS
TOTAL_READS     MAP     EXONIC  INTRONIC        INTERGENIC      TRANSCRIPTOME   ANTISENSE
624859097       624859097       456866088       49810657        118182352       506676745
109864594
100.0   100.0   73.1    8.0     18.9    81.1    17.6
```

3.3.2 Example of Annotated BAM

```
$ samtools view /path/to/output/02.count/SS200000135TL_D1.Aligned.sortedByCoord.out.merge.q10.
dedup.target.bam | grep GE:Z | head -3
E100026571L1C002R00703943265    1040    1       3082766 255     11M132671N89M   *       0
0       CTGCTGCAGCTTTTTTTTCTTTGAGATTTATTTTTATGCTATGTGTATGGGTATTTTGCCTGCATATATGTCTATGCAC-
CATGTGTGTGCAGTGCTTGAG       FFFFFECGFDCFGDGDFEE@EEGIBFGGCGFFGACGFCGFFDGDGFFFFFFEGCDFCGFF-
GG@FFF=EFFDGGGGGFDGFFFGGGFGFFGGGFFGGGDFG    NH:i:1  HI:i:1     AS:i:88 nM:i:0  Cx:i:7767
Cy:i:18052      UR:Z:7AE49      XF:i:0  GE:Z:Xkr4       GS:Z:-  UB:Z:79E49
E100026571L1C006R01702839878    16      1       3108680 255     11M187430N89M   *       0
0       GTCCTTTTTTTTTTTTTTTTTTCAAGTTATTCCCTCCATTTCCTCTGGTTCTGTGTTTATGACCAATTGCAAATTG-
TAGAAAATTTGCATGTGTACTGAAC       5;E4FGCGGGGEGFDFFGGG>GFGFDGBFDFDF:EGEFFGFGGFGGGGFFGGGGFGGF-
GFCFGFFGGGGGFFGGFGGGGGGGGFGGGFGFGBGFGGEFFD    NH:i:1  HI:i:1     AS:i:88 nM:i:0  Cx:i:7159
Cy:i:12715      UR:Z:88DB5      XF:i:1  GE:Z:Xkr4       GS:Z:-
E100026571L1C002R04102972006    16      1       3110829 255     18M211964N82M   *       0
0       ATTGTTTTTTTTTTTTTTTTTTCATGGGAGCTAAAAAATGTTTAATTGTTTGAATAAGAAAAATGTTTCTGATCAAATGTCTCATA-
CAGCTCATATAAAAT       9*E/GGGGGGGGGGGGGGGGGFFDGGFGGG7GGFGGGGGGGGGGGGGGGGFGGFGGGGGGGGGGGFGGGG-
FGGGGGGGGGHGGGGGGGGGGGGGGGGG    NH:i:1  HI:i:1     AS:i:88 nM:i:0  Cx:i:11283      Cy:i:12647
UR:Z:887BF      XF:i:1  GE:Z:Xkr4       GS:Z:-
```

3.3.3 Example of count Gene Expression Matrix

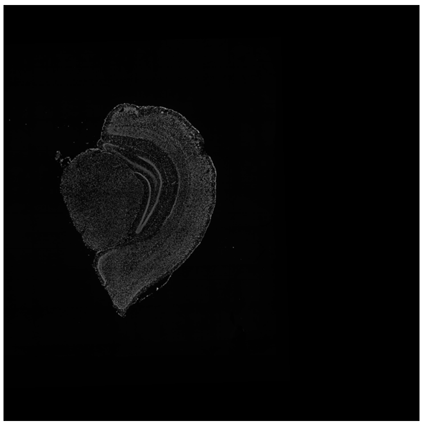
```
$ h5dump -n /path/to/output/02.count/SS200000135TL_D1.raw.gef
HDF5 “/path/to/output/02.count/SS200000135TL_D1.raw.gef” {
FILE_CONTENTS {
 group      /
 group      /geneExp
 group      /geneExp/bin1
 dataset    /geneExp/bin1/expression
 dataset    /geneExp/bin1/gene
 }
}
$ h5dump -d /geneExp/bin1/expression  /path/to/output/02.count/SS200000135TL_D1.raw.
gef | head -15
HDF5 “/path/to/output/02.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/02.count/SS200000135TL_D1.raw.gef |
head -20
HDF5 “/path/to/output/02.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,
         133
      },
   (1): {
```

3.3.4 Example of count Sampling File

```
$ head -8 /path/to/output/02.count/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/03.register/7_result/SS200000135TL_D1_regist`.tif

3.5 tissueCut

3.5.1 Statistical Report for Tissue Covered Region

```
$ cat /path/to/output/04.tissuecut/tissuecut.stat
############## Tissue Statistic Analysis with Stain Image ############
Contour_area: 88336939
Number_of_DNB_under_tissue: 35918358
Ratio: 40.66%
Total_gene_type: 24296
MID_counts: 87120944
Fraction_MID_in_spots_under_tissue: 83.20%
Reads_under_tissue: 657982650
Fraction_reads_in_spots_under_tissue: 77.85%

binSize=1
Mean_reads_per_spot: 18.319
Median_reads_per_spot: 12.0
Mean_gene_type_per_spot: 1.7
Median_gene_type_per_spot: 1.0
Mean_Umi_per_spot: 2.426
Median_Umi_per_spot: 2.0

binSize=50
Mean_reads_per_spot: 18371.193
Median_reads_per_spot: 16481.5
Mean_gene_type_per_spot: 1131.942
Median_gene_type_per_spot: 1098.0
Mean_Umi_per_spot: 2432.459
Median_Umi_per_spot: 2246.0

binSize=100
Mean_reads_per_spot: 72369.407
Median_reads_per_spot: 65597.0
Mean_gene_type_per_spot: 3043.033
Median_gene_type_per_spot: 3038.0
Mean_Umi_per_spot: 9582.154
Median_Umi_per_spot: 8820.5

binSize=150
Mean_reads_per_spot: 160483.573
Median_reads_per_spot: 146299.5
Mean_gene_type_per_spot: 4843.486
Median_gene_type_per_spot: 4969.5
Mean_Umi_per_spot: 21249.011
Median_Umi_per_spot: 19677.0
binSize=200
Mean_reads_per_spot: 281189.167
Median_reads_per_spot: 258306.0
Mean_gene_type_per_spot: 6342.139
Median_gene_type_per_spot: 6640.5
Mean_Umi_per_spot: 37231.173
Median_Umi_per_spot: 34612.5
```

3.5.2 Example of Gene Expression Matrix for Tissue Covered Region

```
$ h5dump -n /path/to/output/04.tissuecut/SS200000135TL_D1.tissue.gef
HDF5 “/path/to/output/04.tissuecut/SS200000135TL_D1.tissue.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/04.tissuecut/SS200000135TL_D1.
tissue.gef | head -15
HDF5 “/path/to/output/04.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 { ( 60922074 ) / ( 60922074 ) }
   DATA {
   (0): {
         5426,
         2662,
         4
      },
   (1): {
$ h5dump -d /geneExp/bin1/gene /path/to/output/04.tissuecut/SS200000135TL_D1.tissue.
gef | head -20
HDF5 “/path/to/output/04.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 { ( 24268 ) / ( 24268 ) }
   DATA {
   (0): {
         “0610005C13R
ik\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000”,
         0,
         23
      },
   (1): {
```

3.5.3 Example of Gene Expression Matrix for Maximum Area Enclosing Rectangle

```
$ h5dump -n /path/to/output/04.tissuecut/SS200000135TL_D1.gef
HDF5 "/path/to/output/04.tissuecut/SS200000135TL_D1.gef" {
FILE_CONTENTS {
 group      /
 group      /geneExp
 group      /geneExp/bin1
 dataset    /geneExp/bin1/expression
 dataset    /geneExp/bin1/gene
group      /geneExp/bin10
 dataset    /geneExp/bin10/expression
 dataset    /geneExp/bin10/gene
 group      /geneExp/bin100
 dataset    /geneExp/bin100/expression
 dataset    /geneExp/bin100/gene
 group      /geneExp/bin20
 dataset    /geneExp/bin20/expression
 dataset    /geneExp/bin20/gene
 group      /geneExp/bin200
 dataset    /geneExp/bin200/expression
 dataset    /geneExp/bin200/gene
 group      /geneExp/bin50
 dataset    /geneExp/bin50/expression
 dataset    /geneExp/bin50/gene
 group      /geneExp/bin500
 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
 }
}
$ h5dump -d /stat/gene /path/to/output/04.tissuecut/SS200000135TL_D1.gef | head -20
HDF5 "/path/to/output/04.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 { ( 24606 ) / ( 24606 ) }
   DATA {
   (0): {
         "Gm42418",
         5851160,
         60.11
      },
   (1): {
```

3.6 saturation

3.6.1 Example of Sequence Saturation File

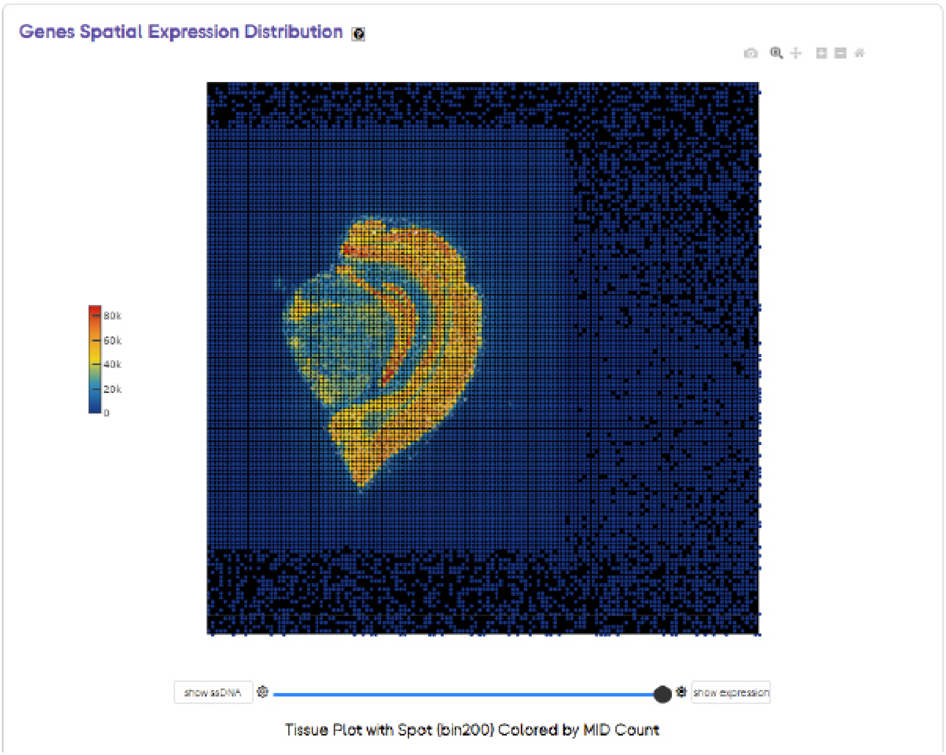
```
$ head /path/to/output/06.saturation/sequence_saturation.txt
#sample bar_x bar_y1 bar_y2 bar_umi bin_x bin_y1 bin_y2 bin_umi
0.05 21857012 0.2451227 1 16499361 21857012 0.2706192 2851 6616
0.1 43714024 0.3835907 1 26945730 43714024 0.4051051 3822 10805
0.2 87428049 0.5359986 1 40566739 87428049 0.5518293 4724 16267
0.3 131142073 0.6192704 1 49929674 131142073 0.631755 5160 20021
0.4 174856098 0.6722583 1 57307641 174856098 0.6826324 5483 22980
0.5 218570122 0.7092118 1 63557618 218570122 0.7181397 5762 25486
0.6 262284147 0.7365943 1 69087139 262284147 0.7444621 5939 27704
0.7 305998171 0.757775 1 74120410 305998171 0.7648404 6068 29722
0.8 349712196 0.7746872 1 78794642 349712196 0.7811282 6188 31596
```

3.7 report

3.7.1 Example of Statistical Summary Report 3.7.2HTML Report

```
$ head /path/to/output/new_final_result.json
{
    “version”: “version_v2”,
    “1.Filter_and_Map”: {
        “1.1.Adapter_Filter”: [
            {
                “Sample_id”: “E100026571_L01_trim_read”,
                “getCIDPositionMap_uniqCIDTypes”: “645.78M”,
                “total_reads”: “1.0G”,
                “fixed_sequence_contianing_reads”: “0.0(0.00%)”,
                “mapped_reads”: “845.17M(84.33%)”,
```

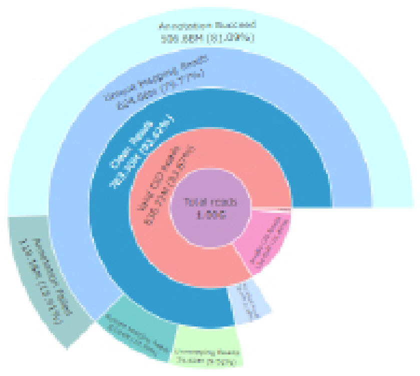
3.7.2 HTML Report


Genes Spatial Expression Distribution
80k
60k
40k
20k
0
show ssDNA
show expression
Tissue Plot with Spot (bin200) Colored by MID Count

1.00G	37.23K	6.34K
Total Reads	Mean MID per Bin200	Mean Gene per Bin200

Sunburst

Total Reads	1.00G
Valid CID Reads	836.72M
Clean Reads	783.30M
Reads Mapped to Genome	708.70M



Sequencing

Total Reads	1.00G
Rate of Q30 Bases in CID	91.74%
Rate of Q30 Bases in MID	89.45%
Rate of Q30 Bases in Seq	91.08%

RNA Mapping

Clean Reads	783.30M
Unique Mapping Reads	624.86M
Multiple Mapping Reads	83.84M
RNA Unmapping Reads	74.60M

Annotation

Exonic	456.87M
Intronic	49.81M
Intergenic	118.18M
Transcriptome	506.68M
Antisense	109.86M

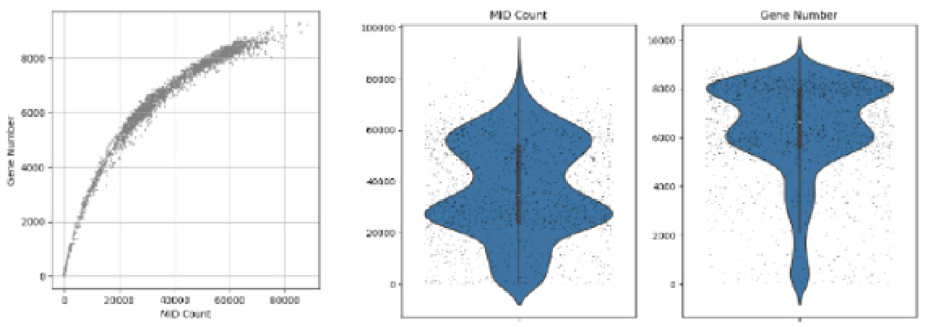
TissueCut Total Stat

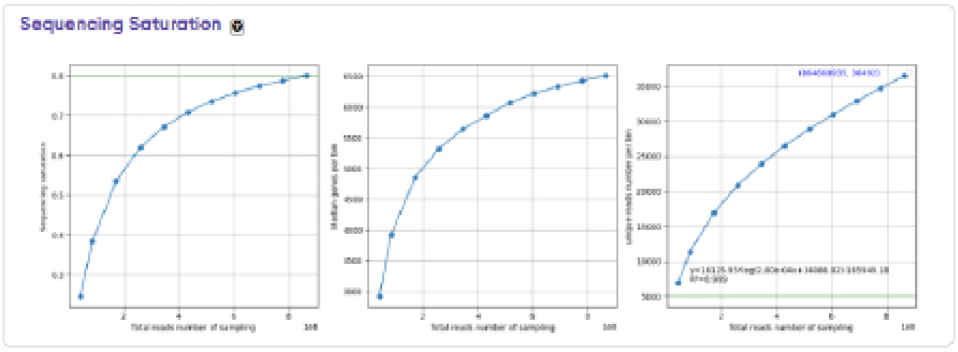
Contour Area	88.34M
Number of DNB Under Tissue	35.92M
Ratio	40.66%
Total Gene Type	24.30K
MID Under Tissue	87.12M
Fraction MID in Spots Under Tissue	83.20%
Reads Under Tissue	657.98M
Fraction Reads in Spots Under Tissue	77.85%

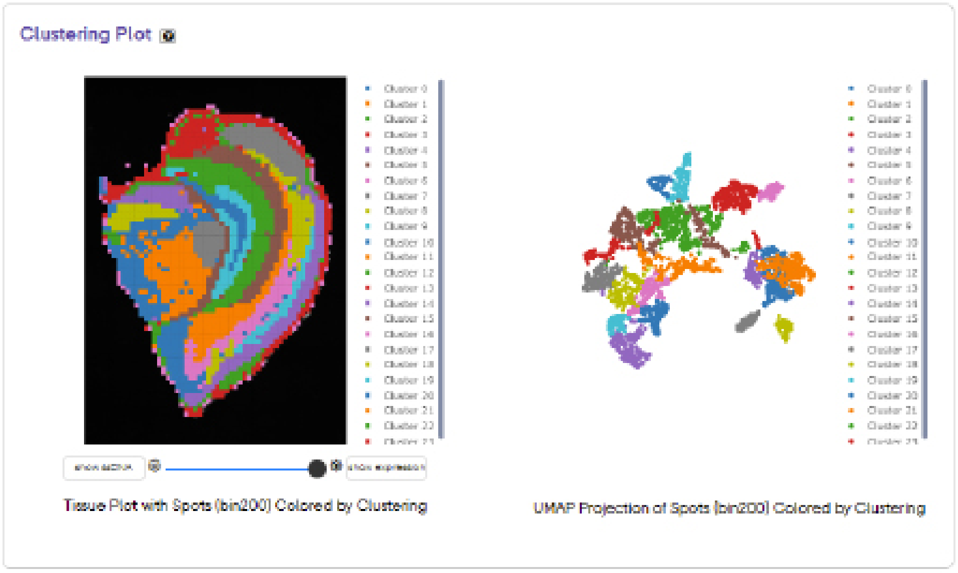
TissueCut Bin Stat

Bin Size	Mean Reads	Median Reads	Mean Gene Type	Median Gene Type	Mean MID	Median MID
50	18.37K	16.48K	1.13K	1.10K	2.43K	2.25K
100	72.37K	65.60K	3.04K	3.04K	9.58K	8.82K
150	160.48K	146.30K	4.84K	4.97K	21.25K	19.68K
200	281.19K	258.31K	6.34K	6.64K	37.23K	34.61K

Bin 200







Tissue Plot with Spots (bin200) Colored by Clustering

UMAP Projection of Spots (bin200) Colored by Clustering

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

http://www.stomics.tech/

Email:support@stereomics.com