时空组学
STOmics

STOmics 分析流程软件包
使用手册

软件版本号：V4.1.0
说明书版本号：A2

版本历史

说明书版本： A0
试剂盒版本： V 1.0
修订日期： 2021 年 11 月
修订内容： 初始版本

说明书版本： A1
试剂盒版本： V 2.1.0
修订日期： 2021 年 12 月
修订内容：

- **新功能：**添加手动配准功能，手动调参后自动微调矫正；
- **优化：**mapping 性能优化；测序饱和度计算移至 tissuecut 之后，仅计算组织覆盖区域情况；
- **Bug 修复：**修复 mapping 构建索引的问题；修复 register 等待时长问题。

说明书版本： A1.1
试剂盒版本： V 2.1.0
修订日期： 2022 年 1 月
修订内容：

- 补充报错处理说明；
- 更新示例数据展示

说明书版本： A2
试剂盒版本： V 4.1.0
修订日期： 2021 年 4 月
修订内容：

- **新功能：**支持处理显微镜拼接的大图；聚类使用 Stereopy；新的基因表达矩阵文件格式；添加格式转换和辅助 mapping 的工具；
- **优化：**mapping 性能提升；更新 count 基因注释逻辑；升级图像拼接和组织分割模型，得到更准确的拼接结果，改善组织空洞处的分割效果。

提示：请下载最新版说明书，与相应版本的软件使用。

©2022 深圳华大生命科学研究院保留所有权利。

1. 本产品仅用于研究，不用于诊断。

2.本手册上的内容可能全部或部分受到适用的知识产权法的保护。深圳华大生命科学研究院和/或相应权利主体依法拥有其知识产权，包括但不限于商标权、版权等。

3.深圳华大生命科学研究院不授予或暗示使用我们或任何第三方的任何版权内容或商标（注册或未注册）的权利或许可。未经本单位书面同意，任何人不得擅自使用、修改、复制、公开传播、更改、分发或发布本手册的程序或内容，不得使用或利用设计技巧使用或占有本单位或本单位关联方的商标、标识或其他专有信息（包括图像、文本、网页设计或形式）。

4.此处的任何内容都无意于或应被理解为对此处列出或描述的任何产品的性能的任何保证、表达或暗示。适用于本文所列任何产品的任何和所有保证均载于购买该产品所附的适用销售条款和条件。深圳华大生命科学研究院不做任何保证，并在此声明对本文所述任何第三方产品或协议的使用不做任何保证。

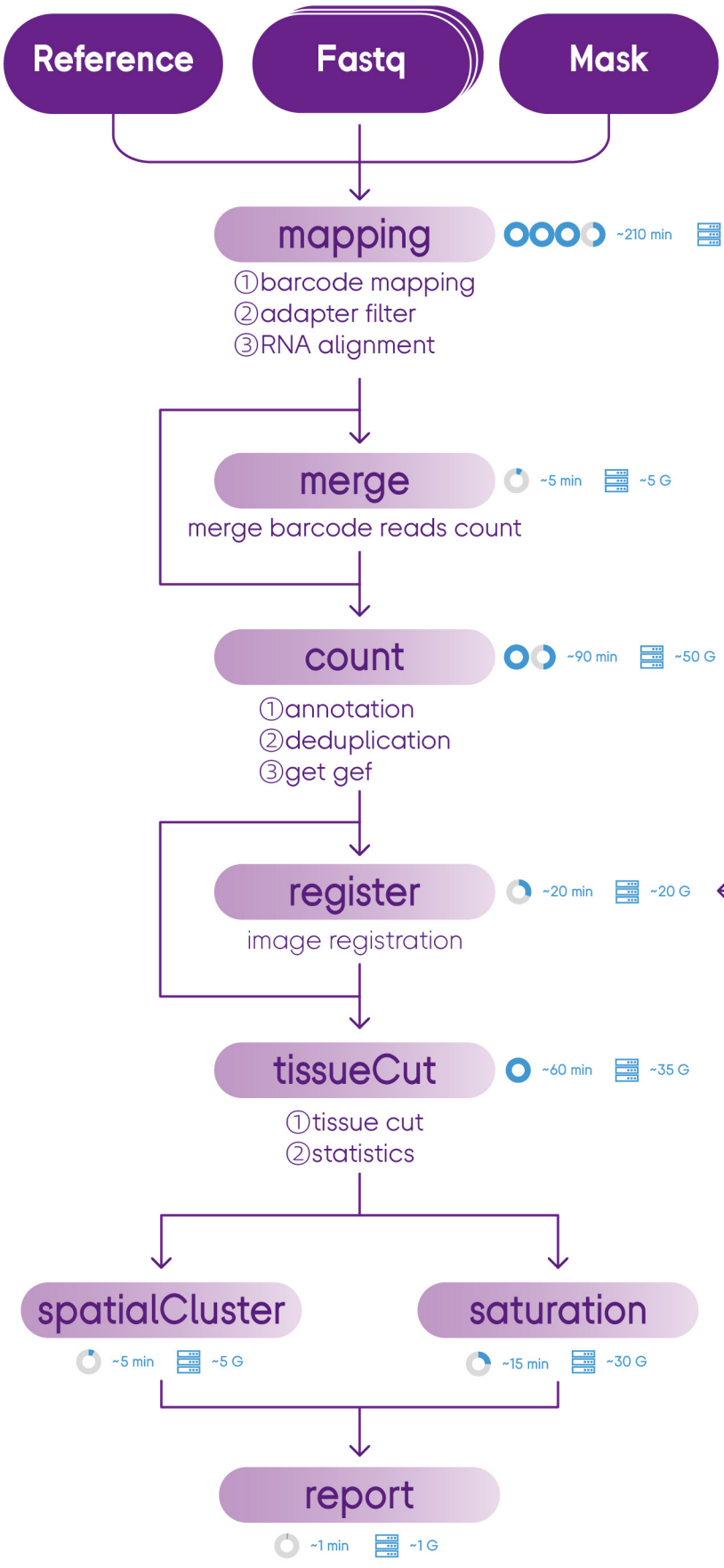
WORKFLOW

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

sequencing report (.summaryReport.html)
bam index (.bam.bai)
statistical report (.stat &.Log.final.out)

reads count (.barcodeReadsCount.txt)

annotated BAM (.bam)
raw matrix (.raw.gef)
statistical report (.stat)

image pyramid (.rpi)

gef
statistical report (.stat)
matrix of tissue covered region (.tissue.gef)

spatial cluster (.h5ad)
saturation plot (.png)

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

目录

第一章
STOmics 分析流程软件包

1.1. 软件描述

STOmics分析流程软件包[1](STOmics Analysis Workflow, SAW)整合了多个STOmics基因表达分析工具，用于将测序数据与空间位置信息结合，实现空间重构。Stereo-seq[2]原始测序数据经SAW分析后，得到基因的空间表达矩阵，可用于下游个性化分析。SAW包含八个主要部分：

- **mapping：**将记录于FASTQ[3,4]文件的Stereo-seq原位捕获的测序reads和空间信息对应。将reads与参考基因组比对，并生成排序后的BAM文件；
- **merge（可选）：**合并CID（类似barcode）对应reads数列表。仅应用于多对FASTQ数据合并分析；
- **count：** 读取mapping生成的BAM文件，对比对结果进行基因注释、去重、和基因表达分析；
- **register（可选）：**用于将显微拍照的组织切片影像图与count生成的基因表达矩阵文件（GEF）进行配准；
- **tissueCut：**根据count生成的基因表达矩阵或同时结合register得到的配准图像，识别芯片上的组织覆盖区域，提取对应位置的基因表达矩阵；
- **saturation：**根据count生成的用于抽样统计的文件计算组织覆盖区域的测序饱和度；
- **spatialCluster：**根据tissueCut生成的组织覆盖区域基因表达矩阵做bin200的聚类分析；
- **report：**生成整合了每步分析结果的JSON格式统计报告，以及HTML网页分析报告，展示基因的空间表达分布、关键统计指标、测序饱和度统计图、以及聚类分析结果；

1.2. 系统要求

运行SAW的Linux系统需满足的最低要求包括：

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

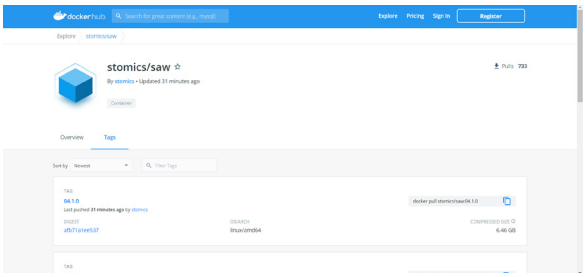
运行SAW需提前安装下列软件其中一个：

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

1.3. SAW 镜像安装

SAW docker 镜像软件包已包含运行软件必须的依赖包，用户可通过 Docker Hub 下载 SAW docker 镜像，以帮助用户在本地快速搭建 SAW 镜像，进行离线分析。**请下载最新的软件并使用与其对应版本的手册。**

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



目前 SAW 的安装和运行支持 Singularity 和 Docker 两种方式 (如需安装 Singularity，请自行安装或参考 1.4 SAW GitHub)，以 4.1.0 版本为例使用教程如下：

1.3.1 Singularity

请注意将红色高亮的入参替换为您的真实路径和数据。

第一步：使用 singularity 拉取镜像 （2 种方法）：

```
$ 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
```

非 root 用户，尤其是 /home/ 目录下磁盘空间不够时，请尝试：

```
$ export SINGULARITY_CACHEDIR=/path/to/build
$ singularity build --sandbox SAW_v4.1.0/ docker://stomics/saw:04.1.0
```

第二步：使用 singularity 运行镜像（3 种方法）：

注意！在运行镜像时，所有涉及目录均需提前挂载。如：输入文件目录 /path/to/data，参考基因组目录 /path/to/genomeDir，输出结果目录 /path/to/output。

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

方法一：启动容器并在容器中执行命令。

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

方法二：启动容器并打开交互式终端，在容器中运行 bash 命令。需执行 exit 退出环境。

```
$ /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
$
```

方法三：通过参数“-B 外部路径 : 容器内路径”挂载用户自定义目录至容器中，并在容器中执行命令。

```
$ 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
$
```

* 注意：这两行属于同一条命令。.

1.3.2 Docker

第一步：使用 docker 拉取镜像：

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

第二步：交互式运行镜像：

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

1.4 SAW Github

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

用户可进入 GitHub 页面查看 **singularity 的安装**、**参考基因组索引构建**、以及 **SAW shell 脚本**。

注意！请在运行 SAW 分析前构建索引。

1.5 SAW 测试数据

测试数据可从 SAW GitHub 页面获取。测试结果可参考 第三章 测试数据结果展示，测试使用参考基因组版本为：

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

1.6 SAW 结果文件格式

SAW 结果文件格式介绍参见《STOmics 分析流程结果文件格式说明》。

第二章
SAW 工具和命令行参数说明

2.1 mapping

Stereo-seq 原始测序数据通过 SAW 软件中的 **mapping** 工具，将存储在 FASTQ 文件中的原始测序 reads 的 CID 与 STOmics 基因表达芯片 Mask 文件记录的 CID- 坐标键值对进行匹配。根据 Mask 文件的记录，为 CID 能够匹配的 reads 添加坐标信息。经过 CID 匹配的 reads 为具有有效 CID 的 **mRNA reads （Valid CID Reads）**。Valid CID Reads 经过滤后，得到 **Clean Reads**。mapping 工具将 **Clean Reads** 与相应物种的参考基因组进行比对，并统计比对结果 BAM[7] 文件。

运行 mapping 所需输入文件包括：

- Stereo-seq 原始测序数据 FASTQ 文件 **（.fq.gz）**
- STOmics Chip Mask 文件 **（.h5）**
- 参考基因组索引文件
- bcPara 文件 **（.bcPara）**，内容参见表 **2-2**

~1G reads 运行时间：~3.5 h，内存：~60G

小提示：如您需要预估 SN 芯片 CID 数量以及预估运行 mapping 大致所需内存，请参考 2.9.1 CIDcount 获取更多信息。

2.1.1 命令行参数

由于mapping工具封装STAR[8]功能，故其可用参数比下表列出的更多。

表 2-1 mapping 命令行参数

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.

表 2-2 mapping --stParaFile 命令行参数

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 示例

准备 **mapping --stParaFile** 输入文件 **{lane}.bcPara**，内容如下：

注意！ {SN} 指代 STOmics 基因表达芯片 SN 编号（如，SS200000135TL_D1），{lane} 指代测序 FASTQ lane 编号（如，E100026571_L01）（下同）。

```
$ 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
```

运行 **mapping** 程序

```
$ 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
```

如遇到如下错误信息，可从命令中删除 --readNameSeparator \" \":

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

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

2.1.3 输出文件

mapping 成功运行完成后，输出文件如下：

```
$ 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
```

若一个样本产生多个测序 FASTQ 文件，需要对每组 FASTQ 文件进行 **mapping** 分析，输出文件如下（以 2 对 FASTQ 为例）：

```
$ 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 (可选)

SAW 的 **merge** 工具用于合并多个 **mapping** 分析结果。

运行 **merge** 所需输入文件包括：

- **mapping** 输出的 CID 对应 reads 数列表（**.txt**）

~1G reads 2 lanes 运行时间：~5 min，内存：~5G

2.2.1 命令行参数

表 2-3 merge 命令行参数

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 示例

```
$ 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
```

* 注意：我们使用反斜杠“\”来作为多行命令的换行符。

2.2.3 输出文件

对多组测序数据进行 **mapping** 分析并进行 merge 处理后，输出结果如下：

```
$ 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 中的 **count** 工具对 **mapping** 的比对结果中的 **Uniquely Mapped Reads** 进行基因注释。通过统计每个注释基因的 reads 数，综合考虑 CID、基因、和 MID 后进行去重，最后计算每个位置每个基因的表达量水平，生成基因表达矩阵。

运行 **count** 所需输入文件包括：

- **mapping** 输出结果 BAM 文件（**.bam**）
- 基因组注释 GTF/GFF[9,10] 文件（**.gtf / .gff**）

~1G reads 运行时间：~1.5 h，内存：~50G

2.3.1 命令行参数

表 2-4 count 命令行参数

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 示例

```
$ 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
```

* 注意：我们使用反斜杠“\”来作为多行命令的换行符。

多对FASTQ时运行时，(以2对FASTQ为例)：

```
$ 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
```

* 注意：我们使用反斜杠“\”来作为多行命令的换行符。

2.3.3 输出文件

count 成功运行完成后，输出结果如下：

```
$ 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 (可选)

SAW 软件中的 **register** 工具可通过配准算法，根据 track 线信息将上传的显微镜拍照的组织染色影像图与 **count** 输出的基因表达矩阵建立图像与空间表达的映射关系。

运行 **register** 所需输入文件包括：

- **count** 输出的基因表达结果文件（**.raw.gef**）
- 通过 ImageQC 处理的显微镜拍照的组织染色影像图（**.tar.gz**）
- 通过 ImageQC 软件整理后的显微镜拍照影像图图像信息报告（**.json**）

~1G reads 运行时间：~20 min，内存：~20G

2.4.1 命令行参数

表 2-5 register 命令行参数

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 示例

```
$ 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 输出文件

register 输出文件如下：

```
$ 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 的 **tissueCut** 工具可根据 **register** 输出的配准图勾画组织轮廓，并进行组织区域的识别和切割，从而去除组织外区域。如没有显微镜拍照的影像图，**tissueCut** 也可以直接基于基因表达矩阵识别组织区域。tissueCut 提取得到的组织区域基因表达矩阵以 GEF 格式存储。用户可从生成的图像金字塔 RPI 文件使用 python 包 Stereopy[11] 导出 TIFF 或 JPG 格式的配准图像。

如 tissueCut 提取矩阵效果不佳，可使用 Stereopy 进行交互式的 lasso 操作提取矩阵。操作指南参见 Stereopy->Examples->Interactive.

运行 **tissueCut** 所需输入文件包括：

- CID 对应 reads 数列表（**.txt**）
- **count** 输出的基因表达结果文件（**.raw.gef**）
- 显微镜拍照的组织染色影像图结果目录（**可选**）

~1G reads 运行时间：~1 h，内存：~35G

2.5.1 命令行参数

表 2-6 tissueCut 命令行参数

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 示例

有配准图时：

```
$ 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
```

无配准图时：

```
$ 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 输出文件

tissueCut输出结果如下：

有配准图时：

```
$ 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
```

无配准图时:：

```
$ 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 分析流程中的 **spatialCluster** 工具用于对组织覆盖区域的 bin200 的数据点以 Leiden 算法做聚类分析。

运行 **spatialCluster** 所需输入文件包括：

- **tissueCut** 输出组织覆盖区域的 GEF 矩阵（**.tissue.gef**）

~1G reads 运行时间：~5 min，内存：~5G

2.6.1 命令行参数

表 2-7 spatialCluster 命令行参数

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 示例

```
$ 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 输出文件

spatialCluster 输出文件如下：：

```
$ 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 分析流程中的 **saturation** 工具用于计算组织覆盖区域的测序饱和度。

运行 **saturation** 所需输入文件包括：

- **count** 输出计算饱和度所需抽样文件（**.txt**）
- **tissueCut** 输出组织覆盖区域的 GEF 矩阵（**.tissue.gef**）
- **mapping** 输出 CID 比对统计文件（**.stat**）
- **count** 输出注释统计文件（**.stat**）

~1G reads 运行时间：~15 min，内存 ~30G

2.7.1 命令行参数

表 2-8 saturation 命令行参数

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 示例

```
$ 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
```

* 注意:这两行属于同一条命令。

多对 FASTQ 时运行时，（以 2 对 FASTQ 为例）：

```
$ 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
```

* 注意:我们使用反斜杠“\”来作为多行命令的换行符。

2.7.3 输出文件

saturation 输出文件如下：

```
$$ 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 分析流程 **report** 工具可帮助用户整合分析报告，生成 JSON 格式的分析结果统计报告以及 HTML 网页版分析报告。分析报告整合基因的空间表达分布、关键统计指标、测序饱和度图、以及聚类分析结果图。

运行 **report** 所需输入文件包括：

- **mapping** 输出 CID 比对统计文件（**.stat**）、STAR 比对统计文件（**.out**）
- **count** 输出注释统计文件（**.stat**）
- **tissueCut** 输出 GEF 文件（**.tissue.gef**）、组织覆盖区域统计文件（**.stat**）、统计图（**.png**）、图像金字塔文件（**.rpi**）
- **spatialCluster** 输出聚类文件（**.h5ad**）
- **saturation** 输出 bin200 的测序饱和度图（**.png**）

~1G reads 运行时间：~1 min，内存：1G

2.8.1 命令行参数

表 2-9 report 命令行参数

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 示例

有配准图时：

```
$ 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
```

* 注意：我们使用反斜杠“\”来作为多行命令的换行符。

多对FASTQ时运行时，（以2对FASTQ为例）：

```
$ 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
```

* 注意:我们使用反斜杠“\”来作为多行命令的换行符。

无配准图时（以单对FASTQ为例，多对FASTQ同理）：

```
$ 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
```

* 注意:我们使用反斜杠“\”来作为多行命令的换行符。

2.8.3 输出文件

report 输出文件如下：

```
$ 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 其他工具

2.9.1 CIDcount

CIDcount是一个用于计算STOmics Chip Mask文件中CID数量以及粗略预估**mapping**所需内存使用的小工具。

```
$ 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}”
```

CIDcount输出结果示例如下：

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

如果运行**CIDcount**，用户可在mapping所必须的**{lane}.bcPara**文件中添加“**bcNum**”参数，示例如下：

```
$ 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]是一个处理GEF矩阵文件的工具。用户可在SAW中使用此工具进行GEF文件的格式转换和补全，也可以使用其python封装的包**gefpy**[13]进行文件处理。

功能一：GEF转为普通GEM格式

```
$ 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
```

功能二：补全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
```

功能三：GEM转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
```

使用gefpy运行示例，用户可指定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')
```

第三章
测试数据结果展示

用户测试SAW流程时可根据此章节内容作为格式参考。此章节包括测试数据关键步骤的数据统计结果，和关键结果文件的部分示例。

SN: SS200000135TL_D1

*示例中的“...“表示可被省略的日志信息。

3.1 mapping

3.1.1 CID 匹配和过滤统计结果

```
$ 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 参考基因组比对统计

```
$ 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 mapping BAM示例

```
$ 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       GGCTTTTTTTTTTTTTTTTTTTTTTTTTTCTAAATATTGGGTTTTATTAGCACCATGATAACT-
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       CTCGACGGCCATTTTTTTTTTTTTTTTTTTTTGGGTGGTAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA       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 CID对应reads数列表示例

```
$ 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 MID过滤和基因注释结果统计

```
$ 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 注释结果BAM示例

```
$ 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       ATTGTTTTTTTTTTTTTTTTTCATGGGAGCTAAAAAATGTTTAATTGTTTGAATAAGAAAAATGTTTCTGATCAAATGTCTCATA-
CAGCTCATATAAAAT       9*E/GGGGGGGGGGGGGGGGGFFDGGFGGG7GGFGGGGGGGGGGGGGGGGGGFGGFGGGGGGGGGGGFGGGG-
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:-
```

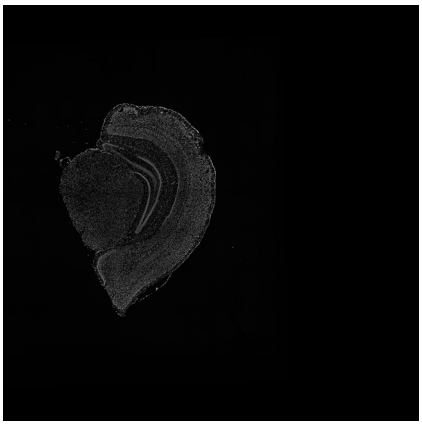
3.3.3 count基因表达文件示例

```
$ 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 count抽样文件

```
$ 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 配准图片image

/path/to/output/03.register/
7_result/SS200000135TL_D1_regist.tif文件

3.5 tissueCut

3.5.1 组织覆盖区域统计分析

```
$ 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 组织覆盖区域基因表达矩阵示例

```
$ 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\000”,
         0,
         23
      },
   (1): {
```

3.5.3 最大外接矩形基因表达矩阵示例

```
$ 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 测序饱和度文件示例

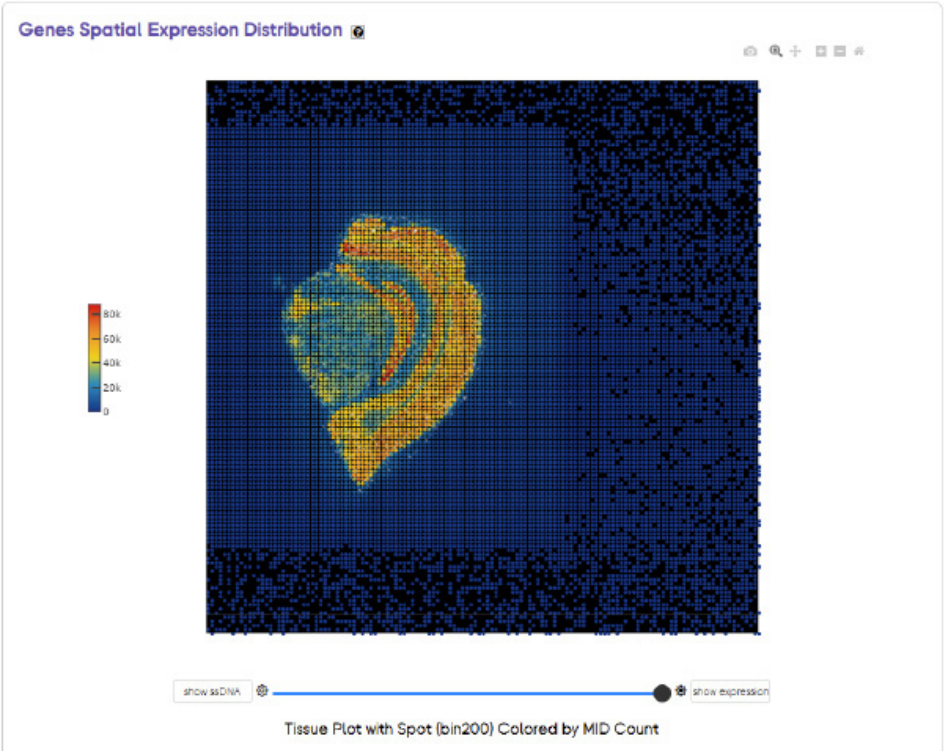
```
$ head /path/to/output/06.saturation/sequence_saturation.txt
#sample bar_x bar_y1 bar_y2 bar_umi bin_x bin_y1 bin_y2 bin_umi
0.05 21857012 0.2451227 1 16499361 21857012 0.2706192 2851 6616
0.1 43714024 0.3835907 1 26945730 43714024 0.4051051 3822 10805
0.2 87428049 0.5359986 1 40566739 87428049 0.5518293 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 分析结果统计报告示例

```
$ 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分析报告



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

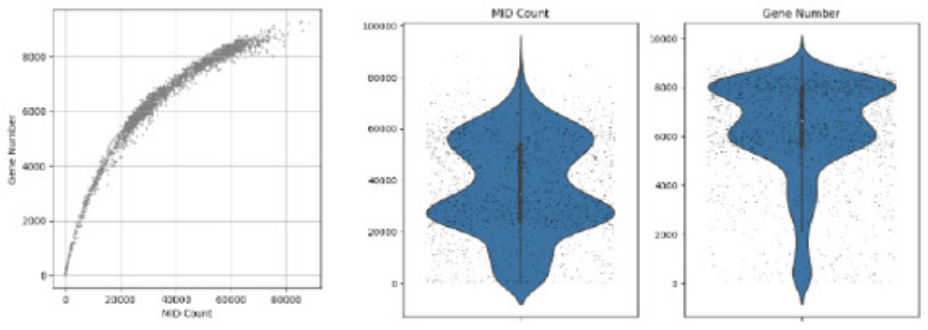
TissueCut Total Stat

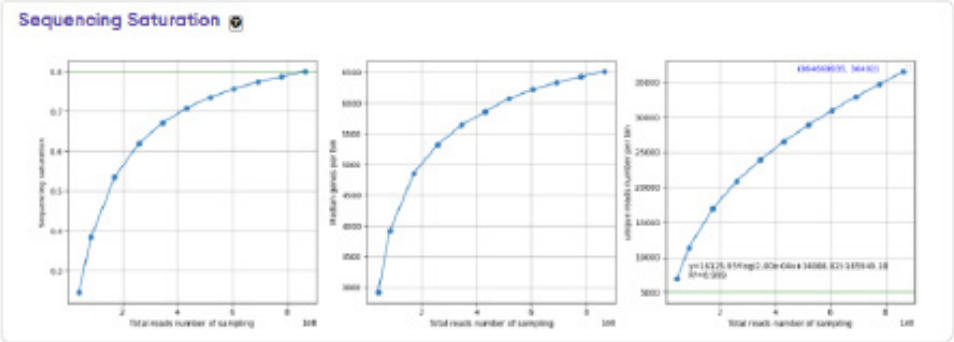
Contour Area	88.34M
Number of DNB Under Tissue	35.92M
Ratio	40.66%
Total Gene Type	24.30K
MID Under Tissue	87.12M
Fraction MID in Spots Under Tissue	83.20%
Reads Under Tissue	657.98M
Fraction Reads in Spots Under Tissue	77.85%

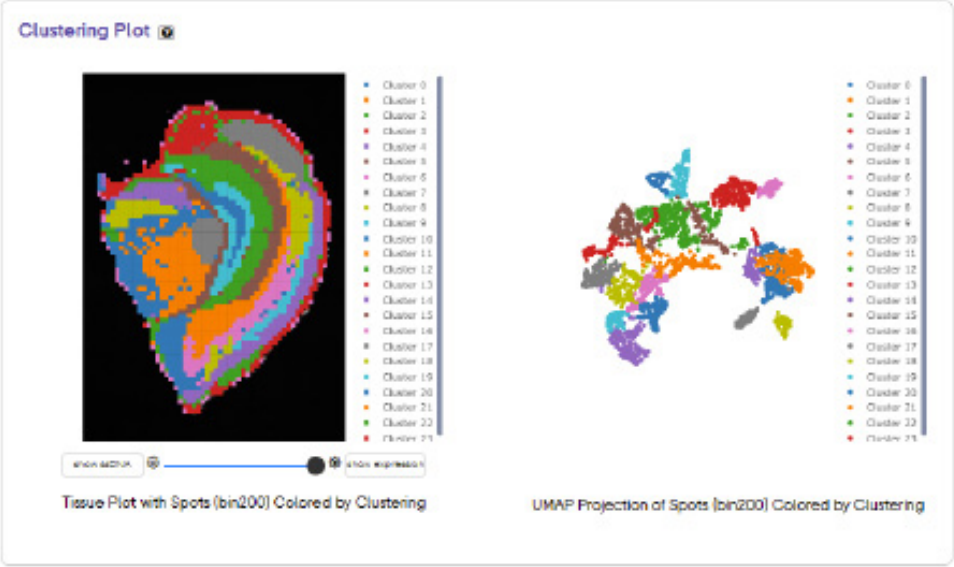
TissueCut Bin Stat

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




Sequencing Saturation


Clustering Plot
Tissue Plot with Spots (bin200) Colored by Clustering
UMAP Projection of Spots (bin200) Colored by Clustering

参考资料

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

联系我们

深圳华大生命科学研究院

网址：https://www.stomics.tech

邮箱：support@stereomics.com