时空组学
STOmics

Stereo-seq 分析流程软件包
使用手册

软件版本号：V6.1

说明书版本号：A7

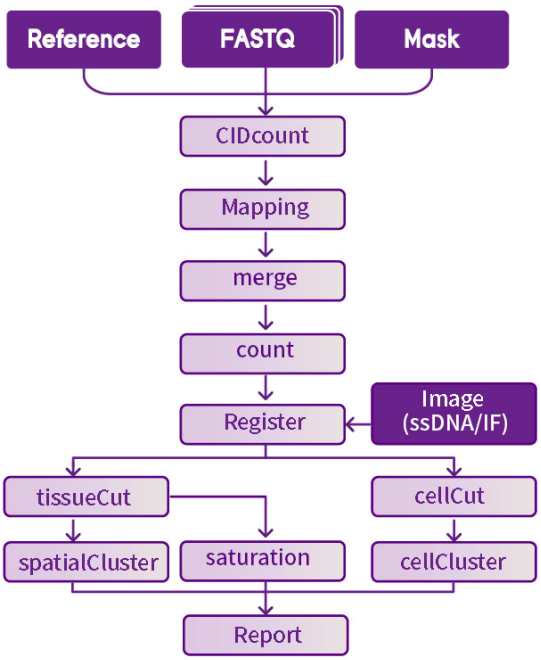
目录

第四章 SAW 常见 Q&A

第一章
SAW 流程简介

1.1. SAW 流程概述

- Stereo-seq分析流程软件包（Stereo-seq Analysis Workflow, SAW）整合了多个Stereo-seq空间组基因表达分析工具，这些工具可用于还原以及可视化测序数据在芯片上的空间表达信息。
- 原始测序数据经SAW分析后，得到可以用于下游分析的空间表达矩阵。为满足更加便捷的数据分析，SAW通常具有13步必要流程，以及其他辅助分析工具。



1.2. SAW 流程版本

当前SAW更新版本V6.1。

- ImageQC: STOmics®显微镜ImageQC软件是一个为评估显微镜图片质量而设计的桌面应用。荧光图片需要通过ImageQC的评估来保证能够满足SAW流程分析的要求。运行SAW>=5.0.0版本要求ImageQC>=1.1.0，以确保可以提供IPR文件来记录图像处理的过程信息。ImageQC的最后一个更新版本为v1.2.0，后续将不会再做更新。
- ImageStudio:ImageStudio是集成了ImageQC功能的升级版本的图像处理桌面应用，包含图像QC、手动拼接、手动组织分割、手动细胞分割四个主要模块。每个模块对应输出结果均可接入SAW进行进一步分析。SAW v6.1版本建议ImageStudio>=v2.1。
- StereoMap:StereoMap是一个支持Stereo-seq分析数据高清交互式可视化的桌面端软件。SAW流程中输出的GEF矩阵、图像RPI和IPR数据、聚类结果等均可在StereoMap中展示。SAW v6.1版本建议StereoMap版本>=v2.1。

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（**下载最新版说明匹配相关版本**）。

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 (如无 Singularity 软件，请自行安装或参考 1.5 SAW GitHub)，

安装的命令如下：

```
$ singularity build SAW_v6.1.sif docker://stomics/saw:06.1.0  ## 方法 1
$ singularity build --sandbox SAW_v6.1/ docker://stomics/saw:06.1.0  ## 方法 2
$ docker pull stomics/saw:06.1.0 ##  方法 3 利用 docker 拉取镜像
$ docker run -d -v /path/to/data:stomics/saw:06.1.0 /bin/sh -c “<shell-command>”
## 方法 4 交互式运行镜像
```

1.4. SAW 运行流程系统要求

运行SAW的Linux系统需满足的最低要求包括：

- **Cpu至少需8核（建议24核）；**
- **运行内存至少128GB（建议256G）；**
- **存储空间至少1TB；**
- **64位 CentOS/RedHat 7.8 或Ubuntu 20.04。**

运行SAW需提前安装下列软件其中一个软件：

- **docker: version >=20.10.8；**
- **singularity: 版本号>= 3.8(推荐使用)。**

1.5. SAW 运行方法

运行方法提供三种 (以 singularity 为例）： （红色字体均为用户需输入的实际路径）

注意！在运行 singularity 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。

```
1 $ export SINGULARITY_BIND="/path/to/data,/path/to/genomeDir,/path/to/output"
```

1. 启动容器并在容器中执行命令。

示例：

```
1 $ /path/to/SAW_v6.1.sif <application> ## 选择 1
1 $ singularity exec /path/to/SAW_v6.1.sif <application> ## 选择 2
```

2. 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。

示例：

```
1 /path/to/SAW_v6.1.sif /bin/bash
2 Singularity> <shell-command>
3 Singularity> exit
```

3. 3. 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。

示例：

```
1 singularity shell -B /path/to/directory/on/the/host-machine:/path/to/directory/
  mounted/in/the/container /path/to/SAW_v6.1.sif
2 Singularity> <shell-command>
3 Singularity> exit
```

1.6. SAW GitHub

SAW GitHub：https://github.com/STOmics/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脚本的详细说明。

注意！请在运行 SAW 分析前构建索引。

1.7. SAW 测试数据

测试数据可从SAW GitHub页面获取。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

第二章
SAW 工具和命令行参数说明

2.1. splitMask（SE 测序步骤）

splitMask： 是通过 SE FASTQ CID 来分割 Stereo-seq 芯片 mask 文件的工具。当从序列中输出 FASTQs 文件时，其分割计数和 FASTQs 文件的分割编号直接相关。

2.1.1. 输入文件

- Stereo-seq 芯片 mask 文件（**.h5**）

SAW 流程支持来自 Stereo-seq 的 Q40 FASTQ 和 Q4 FASTQ 作为测序数据输入。它们之间的区别主要体现在文件大小以及数据和信息的写入方式上。Q40 FASTQ，也叫 PE FASTQ，有一对 read 文件 ,read1 和 read2。Q4 FASTQ，也称为 SE FASTQ，是一种只有一个文件的输出格式，但可以节省文件存储空间。更多细节请参阅 Stereo-seq 文件格式手册。

2.1.2. 命令示例及参数说明

```
$ mkdir /path/to/output/00.splitmask
$ singularity exec SAW_v6.1.sif splitMask \
    /path/to/data/{SN}.barcodeToPos.h5 \ # 必需参数:Stereo-seq 芯片 mask 文件(.h5)
    /path/to/output/00.splitmask \  # 必需参数:输出目录
    8 \ # 必需参数:运行线程数
    16 \ # 可选参数:分割份数
    2_25 # 可选参数
```

注意！请把 {SN} 编号替换成用户自己的 Stereo-Seq 芯片 T 序列编号 (例：SS200000135TL_D1 或 A00135D1)

2.1.3. 运行资源

- 内存占用：3G（1cm*1cm 尺寸芯片数据分析参考值）
- 运行时间：5 分钟（1cm*1cm 尺寸芯片数据分析参考值）

2.1.4. 输出文件

```
/path/to/output/00.splitmask
|-- *.SN.barcodeToPos.bin(* 代表分割编号) (## 通过 CID 分割 Stereo-seq 芯片 mask 文件)
```

2.2. CIDCount

CIDCount： 是用于计算 Stereo-seq 芯片 mask 文件中 CID 数量，以及粗略估算 mapping 过程所需内存的小程序。

2.2.1. 输入文件

- Stereo-seq 芯片 mask 文件（**.h5**）

2.2.2. 命令示例及参数说明

以 PE FASTQ 输入为例，运行 CIDCount 所需文件包括：

```
$ singularity exec SAW_v6.1.sif CIDCount \
  -i /path/to/data/{SN}.barcodeToPos.h5 \ 必需参数:Stereo-seq 芯片 mask 文件(.h5)
  -s {speciesName} \ 可选参数:物种名
  -g {genomeSize} 必需参数 : 基因组大小
```

以 SE FASTQ 输入为例，需要先运行 splitMask，因为每一份拆分后的 mask 文件结果类似，所以用户只需要运行一次 CIDCount 即可。针对大于 1 cm x 1 cm 的芯片，我们推荐用户对每一份拆分 mask 都运行 CIDCount。

大芯片运行建议：对大芯片来说，每一份拆分 mask 文件包含对 CID 数波动可能会较大，所以计算出的预估内存可能会有较明显差异，因此在这种场景下，我们会推荐用户对每一份拆分 mask 文件都做一下 CIDCount 的运算。造成这种波动的原因可能是因为芯片上的 dnb/CID 分布不均。

```
$ singularity exec SAW_v6.1.sif CIDCount \
  -i /path/to/output/00.splitmask/{index}.{SN}.barcodeToPos.bin \   ## 必需参数：Stereo-seq 芯片 mask 文件(.h5)
  -s {speciesName} \ 必需参数:物种名
  -g {genomeSize} 必需参数：基因组大小
```

请注意！{index} 需要替换成拆分芯片文件真实的 index 编号。

2.2.3. 运行资源

- 内存占用：~0.7G（1cm*1cm 尺寸芯片数据分析参考值）
- 运行时间：30s（1cm*1cm 尺寸芯片数据分析参考值）

2.2.4. 输出文件

```
singularity exec SAW_v6.1.sif CIDCount -i SN.barcodeToPos.h5 -s {speciesName} -g 3
  645784920  ##CID 数值
  62  ## 预估比对内存大小
```

2.3. mapping

mapping：每一条 Stereo-seq 序列包含一段 CID 序列，该序列是用于将测序 reads 比对在组织切片上的原始位置的关键信息。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 Stereo-seq 芯片 Mask 文件记录的 CID 坐标键值对进行匹配（允许一位容错）。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，并输出排序后的 BAM 格式比对文件和统计报告。

2.3.1. 输入文件

- Stereo-seq 原始测序数据 FASTQ 文件（**.fq.gz**）
- Stereo-seq 芯片 mask 文件 **(.h5)**
- 参考基因组索引文件
- bcPara 文件（**.bcPara**）

注意！{SN}指代 Stereo-seq 芯片的 SN 编号（例: SS200000135TL_D1）；{lane}指代测序 FASTQ lane 编号（如，E100026571_L01）（下同）。

两个场景：单对和多对 PE FASTQ 的 {lane}.bcPara 文件参数配置如下：

```
$ mkdir /path/to/output/01.mapping  ##option1(单对 PE FASTQ 新建目录)
$ mkdir /path/to/multi_lane_output/01.mapping  ## option2(多对 PE FASTQ 新建目录)
$ vim /path/to/output/01.mapping/{lane}.bcPara  ##option1 (单对 PE FASTQ)创建参数配置文件
$ vim /path/to/multi_lane_output/01.mapping/{lane}.bcPara ##option2 (多对 PE FASTQ)创建参数配置文件
in=/path/to/data/{SN}.barcodeToPos.h5  ## 必需参数:Stereo-seq 芯片 mask 文件(.h5)
in1=/path/to/data/{lane}_read_1.fq.gz  ## 必需参数:Stereo-seq 原始测序数据 FASTQ read1 文件(.fq.gz)
in2=/path/to/data/{lane}_read_2.fq.gz  ## 可选参数:Stereo-seq 原始测序数据 FASTQ read2 文件(.fq.gz)
barcodeReadsCount=/path/to/output/01.mapping/{lane}.barcodeReadsCount.txt  ## option1 (单对 PE FASTQ)必需参数:CID mapping 统计输出
barcodeReadsCount=/path/to/multi_lane_output/01.mapping/{lane}.barcodeReadsCount.txt  ## option2(多对 PE FASTQ)必需参数:CID mapping 统计输出
barcodeStart=0  ## 必需参数:CID 起始位点
barcodeLen=25 ## 必需参数:CID 长度(PE 为 25)
umiStart=25  ## 必需参数:MID 起始位点
umiLen=10 ## 必需参数:MID 长度
mismatch=1 ## 必需参数:mapping 的最大容错碱基数
bcNum=645784920 ## 必需参数:CID 数量,CIDCount 的输出文件第一行
polyAnum=15 ## 可选参数:polyA 长度
mismatchInPolyA=2 ## 可选参数:找寻 polyA 时最大容错碱基数
rRNAremove ## 可选参数:是否过滤 rRN
```

两个场景：含有一个 barcode 且写出 FASTQ 时未做拆分的 SE FASTQ; 2、含有多个 barcode 且写出 FASTQ 时做了拆分的 SE FASTQ 的 {lane}.bcPara 文件参数配置如下：

注意：{index} 需要被替换成拆分 mask 文件的 index 编号，即与 SE FASTQ 文件对应的 index 编号。{idx} 需要换成输入 SE FASTQ 文件的 index 编号。若一个文库同一个测序泳道包含两个 barcode，则 barcode1 的第三条 FASTQ 的 {index} 和 {idx} 分别为 01 和 03。

```
$ mkdir /path/to/output/01.mapping  ## 新建目录
$ vim /path/to/output/01.mapping/{index}.bcPara  ##option1(只含有一个 barcode 且没有拆分 barcode 的 SE FASTQ) 创建参数配置文件
$ vim /path/to/output/01.mapping/{idx}.bcPara  ##option2 (含有多个 barcode 的 SE FASTQ)创建参数配置文件
in=/path/to/output/00.splitmask/{index}.{SN}.barcodeToPos.bin  ## 必需参数:Stereo-seq 芯片分割后的 mask 文件(.bin)
in1=/path/to/data/{lane}_{index}.fq.gz  ## 必需参数:Stereo-seq 原始测序数据 FASTQ 文件(.fq.gz)(FASTQ files for different barcode usually stores in different directory, so /path/to/data/ might change to /path/to/data/{barcode_n})
```

```
barcodeReadsCount=/path/to/ouptut/01.mapping/{index}.barcodeReadsCount.txt  ##option1(只含有一个 barcode 且没有拆分 barcode 的 SE FASTQ)必需参数:CID mapping 统计输出
barcodeReadsCount=/path/to/ouptut/01.mapping/{idx}.barcodeReadsCount.txt  ##option2 (含有多个 barcode 的 SE FASTQ)必需参数:CID mapping 统计输出
barcodeStart=0  ## 必需参数:CID 起始位点
barcodeLen=24  ## 必需参数:CID 长度(SE 为 24)
umiStart=25  ## 必需参数:MID 起始位点
umiLen=10  ## 必需参数:MID 长度
mismatch=1  ## 必需参数:mapping 的最大容错碱基数
bcNum=38284877  ## 必需参数:CID 数量
polyAnum=15 ## 可选参数:polyA 长度
mismatchInPolyA=2  ## 可选参数:找寻 polyA 时最大容错碱基数
rRNAremove ## 可选参数:是否过滤 rRNA
```

若考虑 rRNA 对实验的影响，在后续步骤中对其统计和过滤，需在参考基因组中添加 rRNA 的相关信息。操作主要包含两步：1）将 rRNA 信息添加到 FASTA 文件中，并且在染色体序号列上增加 '_rRNA' 的后缀标识，例如 '1_rRNA'，与原始参考基因信息区分开来；2）根据修改后的参考基因组文件构建索引。

请注意重新构建参考基因组索引，使用与之前相同的命令输入。从 SAW v6.1 开始，SAW 对参考基因组索引和比对算法进行了重构，以获得更高效的计算性能和减少时间成本。

2.3.2. 命令示例及参数说明

PE FASTQ 及 SE FASTQ 运行 mapping 入参示例及说明:

```
$ singularity exec SAW_v6.1.sif mapping \
    --outSAMattributes spatial \ ## 默认参数:转换成空间 bam 文件格式
    --outSAMtype BAM SortedByCoordinate \ ##STAR 参数:将 bam 按照坐标进行排序
    --genomeDir /path/to/genomeDir \ ##STAR 参数:基因组索引路径
    --runThreadN 8 \ ##STAR 参数:运行线程数
    --outFileNamePrefix /path/to/output/01.mapping/{lane}. \  ##STAR 参数:输出文件前缀(SE FASTQ 将 {lane} 替换成 {index})
    --sysShell /bin/bash \ ## 可选参数:shell 文件目录
    --stParaFile /path/to/output/01.mapping/{lane}.bcPara \ ## 必需参数:CID 比对选项参数文件(SE FASTQ 将 {lane} 替换成 {index})
    --readNameSeparator \" \" \  ##STAR 参数:分隔符
    --limitBAMsortRAM 38582880124 \ ##STAR 参数:排序 bam 文件的最大运行内存
    --limitOutSJcollapsed 10000000 \ ##STAR 参数:最大崩溃连接数
    --limitIObufferSize=280000000 \  ##STAR 参数:每个线程最大可获得的 buffer 大小
    --outBAMsortingBinsN 50 \ ##STAR 参数:坐标排序的基因组 bins 的数量
    --outSAMmultNmax -1\  ##STAR 参数:一条 read 写入 bam 文件中的最大比对数,默认全部输出
> /path/to/output/01.mapping/{index}.run.log ## {index} or {idx} depending on the scenario
```

2.3.3. 运行资源

- 内存占用：~67G（1G reads 数据分析参考值）
- 运行时间：~4h（1G reads 数据分析参考值）

2.3.4. 输出文件

场景 1：单对 PE FASTQ 文件输出目录如下：

```
/path/to/output/01.mapping/
├── lane.Aligned.sortedByCoord.out.bam ## 二进制比对 / 映射文件,用于存储序列比对信息
├── lane.CIDMap.stat ##CID 比对的统计报告
├── lane.barcodeReadsCount.txt ##  比对上 CID 的 reads 列表文件
├── lane.bcPara ## 定义 CID 比对选项的参数文件
├── lane.Log.final.out ## mapping 完成后汇总比对统计信息（STAR 输出）
├── lane.Log.out ##STAR mapping 中的主日志文件（STAR 输出）
├── lane.Log.progress.out  ## 报告作业过程统计数据（STAR 输出）
├── lane.run.log  ## 比对模块的日志文件
└── lane.SJ.out.tab ##mapping 过程中拼接接头的检测（STAR 输出）
```

场景 2：多对 PE FASTQ 文件输出目录：将上述目录中的 lane 替换成 lane* 文件名展示每一对的比对结果。
场景 3: 有一个 barcode 或者没有拆分 barcode 的 SE FASTQ 文件输出目录: 将上述目录中的 lane 替换成 {index}* 文件名。
场景 4：有多个 barcode 的 SE FASTQ 文件输出目录：将上述目录中 lane 替换成 {idx}* 文件名。

2.4. merge

merge：用于合并多个 mapping 分析结果。

2.4.1. 输入文件

- mapping 输出映射后的 CID 列表文件 **(.txt)**

2.4.2. 命令示例及参数说明

```
singularity exec SAW_v6.1.sif merge \
    /path/to/data/{SN}.barcodeToPos.h5 \  必需参数:Stereo-seq 芯片 mask 文件(.h5)
   /path/to/multi_lane_output/01.mapping/{lane*}.barcodeReadsCount.txt \ 必需参数:
mapping 步骤产生的 CID 列表文件
    /path/to/multi_lane_output/02.merge/{SN}.barcodeReadsCount.txt 必需参数:merge
CID 列表文件后产生的文件
```

2.4.3. 运行资源

- 内存占用：~5G（1G reads 2 lanes 数据分析参考值）
- 运行时间：5 分钟（1G reads 2 lanes 数据分析参考值）

2.4.4. 输出文件

```
/path/to/multi_lane_output/02.merge
└── {SN}.merge.barcodeReadsCount.txt ## 合并比对上 CID 的 reads 列表文件
```

2.5. count

count: 一个高效、多用途的序列注释模块，它标记每个 reads 的基因组特征，并输出整体的注释统计信息。通过量化被注释的基因测序 reads，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。SAWcount 通常对 mapping 比对结果中的 Uniquely Mapped Reads 进行基因注释。从 SAW v5.1.3 开始，count 允许重新利用部分 Multi-Mapped Reads 进行量化 (--multi_map)，在 SAW 流程中，基因表达水平包括内含子和外显子的 MID 数量的总和。为了满足下游分析区分不同基因功能区分析的需求，count 会将外显子的 MID 计数单独输出。

2.5.1. 输入文件

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

2.5.2. 命令示例及参数说明

单对 PE FSATQ 及 SE FASTQ 运行 count 参数如下：

```
$ mkdir -p /path/to/output/03.count
$ geneExp=/path/to/output/03.count/{SN}.raw.gef
$ saturationSamplingFile=/path/to/output/03.count/{SN}_raw_barcode_gene_exp.txt
$ singularity exec SAW_v6.1.sif count \
    -i /path/to/output/01.mapping/{lane}.Aligned.sortedByCoord.out.bam \ ## 必需参数: mapping 比对的 bam 文件
    -o /path/to/output/03.count/{SN}.Aligned.sortedByCoord.out.merge.q10.dedup.target.bam \  ## 必需参数:去重合并的 bam 输出文件
    -a /path/to/reference/genes.gtf \  ## 必需参数:基因组注释文件
    -s /path/to/output/03.count/{SN}.Aligned.sortedByCoord.out.merge.q10.dedup.target.bam.summary.stat \  ## 必需参数:bam 统计文件
    -e /path/to/output/03.count/{SN}.raw.gef \  ## 必需参数:设置表达量输出的文件
    --umi_len 10 \ ## 必需参数:MID 的长度
    --sat_file ${saturationSamplingFile} \  ## 可选参数:设置测序饱和度输出的文件
    --sn {SN} \  ## 必需参数:芯片号
    --umi_on \  ## 可选参数:纠正 MID
    --save_lq \  ## 可选参数:保存低质量 reads
    --save_dup \  ## 可选参数:保存重复 reads
    -c 24 \ ## 可选参数:cpu 使用核数
    -m 128  ## 可选参数:内存使用数
    --multi_map  ## 可选参数:处理 multi-mapped reads 可添加该参数
```

注意！当有多对 PE FASTQ 运行 count 时，只需将上述参数中 /path/to/output/ 改成 /path/to/multi_lane_output/，{lane} 改成 {lane*} 即可。

2.5.3. 运行资源

- 内存占用：~45G（1G reads 数据分析参考值）
- 运行时间：~30 分钟（1G reads 数据分析参考值）

2.5.4. 输出文件

```
/path/to/output/03.count
├── SN.Aligned.sortedByCoord.out.merge.q10.dedup.target.bam ## 按坐标排序的带注释的BAM 文件
├── SN.Aligned.sortedByCoord.out.merge.q10.dedup.target.bam.csi ## 带注释的 BAM 索引文件
├── SN.Aligned.sortedByCoord.out.merge.q10.dedup.target.bam.summary.stat##count 统计文件
├── SN_raw_barcode_gene_exp.txt ## 一个记录坐标、基因、MID 和计数信息的，以空格分隔的列表
└── SN.raw.gef ##HDF5 格式的基因表达文件
```

注意！当使用添加了 rRNA 信息的参考基因组进行 mapping，并且启用了 rRNAremove 参数，*.bam.summary.stat 结果文件中的统计字段 PASS FILTER 将不包含匹配上 rRNA 参考序列的片段数目。

2.6. register

register：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。

染色图像包括：DAPI/ssDNA 图像用于细胞核的染色； IF（免疫荧光）图像用于蛋白质的染色。

SAW 的 register 包括四个主要模块：拼接、组织分割、细胞分割和配准。拼接是指把带有重叠部分的显微镜图像拼成全景图像（如果输入文件已经是全景图像则跳过此步骤）。组织分割和细胞分割可分别对组织和细胞覆盖区域进行识别，并生成二值化掩模图。配准可对拼接图像和表达矩阵进行配准，同时使用相同参数将组织和细胞分割二值化图与矩阵配准，图像拼接和分割模块可以和配准模块分开运行。在处理 DAPI 和 mIF(multiple immunofluorescence images) 结合的多图场景时，输出配准后的 DAPI 和 mIF 图像文件。其中，IF 图像的组织分割为阈值式分割结果，细胞分割为 DAPI 组织分割区域、IF 阈值分割、DAPI 细胞分割的交集结果。

目前支持手动处理 QC 失败数据。组织或细胞分割图像将采用来自 ImageStudio 的手动处理结果（接受 QC 失败和成功）。请查看 GitHub 上的手动教程。

细胞分割是最耗时的部分。可以选择跳过细胞分割来运行 rapidRegister，但仍然可以运行拼接、组织分割和配准。可查看 3.2 rapidRegister 以获取有关 rapidRegister 的更多信息。

2.6.1. 输入文件

- count 输出的基因表达结果文件（**.raw.gef**）
- 显微组织染色图像文件（**.tar.gz**）由 ImageStudio 处理。register 支持 ImageStudio v1&v2 以及 ImageQC 版本 >= 1.1.0 的质控输出。
- 通过 ImageStudio 软件整理后的显微镜拍照影像图图像信息报告（**.ipr**）

2.6.2. 命令示例及参数说明

场景 1：处理来自 ImageStudio 的图像。执行拼接、组织分割、细胞分割，并与基因表达矩阵配准。

```
$ image=/path/to/data/image
$ image4register=$(find ${image} -maxdepth 1 -name {SN}*.tar.gz | head -1)
$ imageStudio=$(find ${image} -maxdepth 1 -name {SN}*.ipr | head -1)
$ mkdir -p /path/to/output/04.register
$ singularity exec SAW_v6.1.sif register \
    -i ${image4register} \   ## 必需参数:图像 QC 后产生的 tar.gz 文件
```

```
    -c ${imageStudio} \   ## 必需参数:图像 QC 后产生的 IPR 文件
    -v /path/to/output/03.count/{SN}.raw.gef \ ## 可选参数:输入基因表达矩阵 gef 文件
    -o /path/to/output/04.register   ## 必需参数:配准步骤输出文件目录
```

场景 2：处理来自 ImageStudio 的图像。执行拼接、组织分割和细胞分割，而不与基因表达矩阵进行配准。

```
$ image=/path/to/data/image
$ image4register=$(find ${image} -maxdepth 1 -name {SN}*.tar.gz | head -1)
$ imageStudio=$(find ${image} -maxdepth 1 -name {SN}*.ipr | head -1)
$ mkdir -p /path/to/output/04.register
$ singularity exec SAW_v6.1.sif register \
    -i ${image4register} \   ## 必需参数:图像 QC 后产生的 tar.gz 文件
    -c ${imageStudio} \   ## 必需参数:图像 QC 后产生的 IPR 文件
    -o /path/to/output/04.register   ## 必需参数:输出文件目录
```

场景 3：处理来自场景 2 register 基因表达矩阵处理过的图像。

```
$ imageStudio=$(find /path/to/output/04.register -maxdepth 1 -name {SN}*.ipr |
head -1)
$ mkdir -p /path/to/output/04.register
$ singularity exec SAW_v6.1.sif register \
    -i /path/to/output/04.register \   ## 必需参数:场景 2 的输出文件目录
    -c ${imageStudio} \   ## 必需参数:场景 2 输出的 IPR 文件
    -v /path/to/output/03.count/{SN}.raw.gef \ ## 必需参数:图像 QC 后产生的 IPR 文件
    -o /path/to/output/04.register   ## 必需参数:输出文件目录
```

2.6.3. 运行资源

- 内存占用：~20G（1cm*1cm 尺寸芯片数据分析参考值）
- 运行时间：~120 分钟（1cm*1cm 尺寸芯片数据分析参考值）

2.6.4. 输出文件

场景 1 和场景 3 的 register（如果 -i and -o 是相同的）输出文件如下：

```
/path/to/output/04.register
├── <stainType>_fov_stitched_transformed.tif   ##TIFF 格式的已经与 track 线模板预配准的拼
接全图
├── SN_<chipType>_date_time_version.ipr ##IPR 格式图像处理记录文件
└── SN_tissue_bbox.csv   ## 配准全图的尺寸
```

场景 2 的 register 输出文件：

```
/path/to/output/04.register
├── <stainType>_fov_stitched_transformed.tif   ##TIFF 格式的已经与 track 线模板预配准的拼
接全图
└── SN_<chipType>_date_time_version.ipr ##IPR 格式图像处理记录文件
```

场景 3 的 register 输出文件（如果 -i and -o 是两个不同的路径）：

```
/path/to/output/04.register
├── SN_<chipType>_date_time_version.ipr ##IPR 格式图像处理记录文件
└── SN_tissue_bbox.csv  ## 配准全图的尺寸
```

2.7. imageTools

imageTools：是一个专为处理 SAW 图像数据而设计的方便、实用的工具包。imageTools ipr2img（或 ipr2img）是 SAW 里将图像从 IPR 文件“解码”的必备核心工具。 SAW imageTools ipr2img 可以从 IPR 文件输出 TIFF 格式文件，如配准前的 ssDNA 拼接图片、组织分割和细胞分割二值化掩膜图 TIFF 图片，也会对三类图像进行配准。

请查看 3.4 imageTools 其他应用来了解更多关于 imageTools 的信息。

2.7.1. 输入文件

- ImageQC/ImageStudio 处理过的显微组织染色图像文件 **(.tar.gz)**
- register 处理过的图像处理记录文件 **(.ipr)**

2.7.2. 命令示例及参数说明

```
$ image=/path/to/data/image
$ image4register=$(find ${image} -maxdepth 1 -name {SN}*.tar.gz | head -1)
$ imageIPR=$(find /path/to/output/04.register -maxdepth 1 -name {SN}*.ipr | head -1)
$ singularity exec SAW_v6.1.sif imageTools ipr2img \
    -i ${image4register} \  ## 必需参数:图像 QC 产生的 tar.gz 文件
    -c ${imageIPR} \  ## 必需参数:register 或 rapidRegister 运行后产生的 IPR 文件
    -d tissue cell \ ## 必需参数:输出组织分割或细胞分割文件,以空格分割
    -r True \  ## 必需参数:True: 输出配准后的图像 ; False: 输出预配准图像。
    -o /path/to/output/04.register  ## 必需参数:输出文件目录
```

2.7.3. 运行资源

- 内存占用：10G（1cm*1cm 尺寸芯片数据分析参考值）
- 运行时间：5 分钟（1cm*1cm 尺寸芯片数据分析参考值）

2.7.4. 输出文件

如果 -c IPR 的父级目录和 -o 是相同的，则输出文件为：

```
/path/to/output/04.register
├── <stainType>_fov_stitched_transformed.tif  ##TIFF 格式的已经与 track 线模板预配准的拼接全图
├── <stainType>_matrix_template.txt  ## 配准 DAPI/IF 图的 track 线交叉点模版。用于评估配准结果
├── <stainType>_SN_mask.tif  ##TIFF 格式的配准后的 DAPI/IF 细胞分割二值化图
```

```
├── <stainType>_SN_regist.tif  ##TIFF 格式的配准全景图
├── <stainType>_SN_tissue_cut.tif  ##TIFF 格式的 DAPI/IF 全图文件的组织分割结果
├── <stainType>_transform_template.txt  ##<stainType>_fov_stitched_transformed.tif 的 track 线交叉点模板。用于评估拼接结果
├── fov_stitched_transformed.rpi  ## 已经与 track 线模板配准的拼接全图，支持存储多种 TIFF 格式的染色拼接全图
├── SN.rpi  ## 保存配准后的显微拍照全景图、组织边界、以及细胞边界（降采样）的图像金字塔
├── SN_<chipType>_date_time_version.ipr  ##IPR 格式图像处理记录文件
└── SN_tissue_bbox.csv  ## 配准全图的尺寸
```

如果 -c IPR 的父级目录和 -o 是不同的路径，则输出文件为：

```
/path/to/output/04.register
├── <stainType>_fov_stitched_transformed.tif ##TIFF 格式的已经与 track 线模板预配准的拼接全图
├── <stainType>_matrix_template.txt  ## 配准 DAPI/IF 图的 track 线交叉点模版。用于评估配准结果
├── stainType>_SN_mask.tif  ##TIFF 格式的配准后的 DAPI/IF 细胞分割二值化图
├── <stainType>_SN_regist.tif  ##TIFF 格式的配准全景图
├── <stainType>_SN_tissue_cut.tif  ##TIFF 格式的 DAPI/IF 全图文件的组织分割结果
├── <stainType>_transform_template.txt  ##<stainType>_fov_stitched_transformed.tif 的 track 线交叉点模板。用于评估拼接结果
├── SN.rpi  ## 保存配准后的显微拍照全景图、组织边界、以及细胞边界（降采样）的图像金字塔
└── SN_tissue_bbox.csv  ## 配准全图的尺寸
```

2.8. tissueCut

tissueCut：工具可根据 register 和 imageTools 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。

如果 tissueCut 的结果与组织形态不符，有图场景下可在 ImageStudio 进行手动图像组织分割，之后重新运行 register、imageTools 和 tissueCut 进行矩阵提取。无图场景下可在 StereoMap 进行交互式 lasso 操作后接入 SAW 的 lasso 模块（3.6 lasso）来提取表达矩阵，也可以使用 Stereopy 进行交互式的 lasso 操作来提取表达矩阵。操作指南参见 Stereopy->Tutorials->MaxtrixExport 。

2.8.1. 输入文件

- CID 对应 reads 数列表（**.txt**）
- count 输出的基因表达结果文件（**.raw.gef**）
- imageTools ipr2img 配准后的组织分割二值化掩膜图（**.tif, 可选**）

2.8.2. 命令示例及参数说明

场景 1：显微镜染色图经过 register 处理后，运行 tissueCut：

```
$ mkdir -p /path/to/output/05.tissuecut
$ singularity exec SAW_v6.1.sif tissueCut \
    --dnbfile /path/to/output/02.merge/{SN}.merge.barcodeReadsCount.txt \  ## 可选参数:每个 CID 上的 reads 数量列表文件
    -i /path/to/output/03.count/{SN}.raw.gef \  ## 必需参数:基因表达矩阵文件
    -o /path/to/output/05.tissuecut \  ## 必需参数:tissueCut 输出目录
    -s /path/to/output/04.register/<stainType>_SN_tissue_cut.tif \  ## 可选参数:输入 TIFF 格式组织分割二值化掩模图
    --sn {SN} \  ## 必需参数:芯片号
    -O Transcriptomics \  ## 必需参数:组学具体的字符串
    -d ## 必需参数:为 report 步骤必需参数
```

根据自定义标签区域的掩膜图像生成标签 GEF：

```
$ label= <label_name>## 自定义标签名字
$ labelMaskFile=$(find /path/to/output/04.register -maxdepth 1 -name *${label}*_tissue_cut.tif)
$ mkdir -p /path/to/output/05.tissuecut/tissuecut_${label}
$ singularity exec SAW_v6.1.sif tissueCut \
    --dnbfile /path/to/output/02.merge/{SN}.merge.barcodeReadsCount.txt \  ## 可选参数:每个 CID 上的 reads 数量列表文件
    -i /path/to/output/03.count/{SN}.raw.gef \  ## 必需参数:基因表达矩阵文件
    -o /path/to/output/05.tissuecut/tissuecut_${label} \  ## 必需参数:tissueCut 输出目录
    -s ${labelMaskFile} \  ## 可选参数:输入 TIFF 格式组织分割二值化掩模图
    -l ${label} \  ## 可选参数:自定义标签名
    --sn {SN} \  ## 必需参数:芯片号
    -O Transcriptomics \  ## 必需参数:组学具体的字符串
    -d ## 必需参数:为 report 步骤必需参数 ”
```

场景 2：无配准图时，运行 tissueCut：

```
$ mkdir -p /path/to/output/05.tissuecut
$ singularity exec SAW_v6.1.sif tissueCut \
    --dnbfile /path/to/output/02.merge/{SN}.merge.barcodeReadsCount.txt \  ## 可选参数:每个 CID 上的 reads 数量列表文件
    -i /path/to/output/03.count/{SN}.raw.gef \  ## 必需参数:基因表达矩阵文件
    -o /path/to/output/05.tissuecut \  ## 必需参数:tissueCut 输出目录
    --sn {SN} \  ## 必需参数:芯片号
    -O Transcriptomics \  ## 必需参数:组学具体的字符串
    -d  ## 必需参数:为 report 步骤必需参数
```

2.8.3. 运行资源

- 内存占用：10G（1G reads 数据分析参考值）
- 运行时间：10 分钟（1G reads 数据分析参考值）

2.8.4. 输出文件

场景 1： 含有 -s 参数，输入了配准后的组织图的输出文件列表
根据相应的组织掩膜图像生成组织 GEF。

```
/path/to/output/05.tissuecut
├── SN.tissue.gef   ##HDF5 格式的基因表达文件。组织 GEF 包括组织覆盖区域的表达信息
├── tissuecut.stat   ## 组织覆盖区域的统计报告
└── tissue_fig   ## 该目录存储组织区域覆盖统计图
```

根据自定义标签区域的掩膜图像生成标签 GEF：

```
/path/to/output/05.tissuecut/tissuecut_<label>
├── SN.<label>.raw.label.gef    ##HDF5 格式的基因表达文件。
├── <label>.tissuecut.stat    ## 组织覆盖区域的统计报告
└── tissue_fig ## 该目录存储组织区域覆盖统计图
```

场景 2：不含 -s 参数，未输入配准后的组织图的输出文件列表：

```
/path/to/output/05.tissuecut
├── 100X100_contour_image.png   ##bin100 表达图
├── bin1_img.tif   ##bin1 表达分布 TIFF 文件
├── bin1_img_tissue_cut.tif   ##bin1 表达图中获取的组织掩模图
├── SN.tissue.gef   ##HDF5 格式的基因表达文件。组织 GEF 包括组织覆盖区域的表达信息
├── tissuecut.stat   ## 组织覆盖区域的统计报告
└── tissue_fig   ## 该目录存储组织区域覆盖统计图
```

2.9. spatialCluster

spatialCluster：使用 Stereopy 进行空间聚类分析。

聚类过程包括 4 个步骤：

（1）组织覆盖区域基因表达数据预处理（对每个基因归一化、对数化、高变基因识别和标准化）；

（2）PCA 降维；

（3）使用 UMAP 计算邻域图并做低维嵌入；

（4）使用 Leiden 算法进行聚类分析。

2.9.1. 命令示例

```
$ mkdir -p /path/to/output/06.spatialcluster
$ singularity exec SAW_v6.1.sif spatialCluster \
    -i /path/to/output/05.tissuecut/{SN}.tissue.gef \  ## 必需参数:组织区域 gef 文件
    -o /path/to/output/06.spatialcluster/{SN}.spatial.cluster.h5ad \  ## 必需参数:输出聚类 h5ad 文件路径
    -s 200 ## 必需参数:bin 的大小选择
```

2.9.2. 运行资源

- 内存占用：~5G（1G reads 数据分析参考值）
- 运行时间：~1 分钟（1G reads 数据分析参考值）

2.9.3. 输出文件

```
/path/to/output/06.spatialcluster
└── SN.spatial.cluster.h5ad  ## 空间聚类结果文件
```

2.10. cellCut

cellCut：是基于从 register 和 imageTools 生成的配准图像提取细胞核表达矩阵的工具。cellCut 在 cellbin GEF 格式下输出表达数据。如果 cellbin 结果满意，可以运行 cellCut。

请查阅 3.1 SAW 工具和命令行参数说明 - 其他应用工具 -cellCut 的其他应用来学习更多关于 cellCut 的知识。

2.10.1. 输入文件

- count 工具输出基因表达矩阵文件（**.raw.gef**）
- register 和 imageTools 工具输出细胞分割的二值化掩膜 TIFF 文件（**.tif**）

2.10.2. 命令示例

```
$ mkdir -p /path/to/output/051.cellcut
$ singularity exec SAW_v6.1.sif cellCut cgef \
    -i /path/to/output/03.count/{SN}.raw.gef \  ## 必需参数:原始 gef 文件
    -m /path/to/output/04.register/<stainType>_SN_mask.tif  \  ## 必需参数:TIFF 格式的配准后的 DAPI/IF 细胞分割二值化图
    -o /path/to/output/051.cellcut/{SN}.cellbin.gef  ## 必需参数:输出细胞分割后 gef 文件
```

2.10.3. 运行资源

- 内存占用：~10G（1G reads 数据分析参考值）
- 运行时间：~2 分钟（1G reads 数据分析参考值）

2.10.4. 输出文件

```
/path/to/output/051.cellcut
└── SN.cellbin.gef  ##HDF5 格式的细胞基因表达文件
```

2.11. cellCluster

cellCluster：使用 Stereopy 进行细胞聚类，该过程与 spatialCluster 类似。

2.11.1. 输入文件

- cellCut 输出 cellbin 基因表达矩阵文件（**.cellbin.gef**）

2.11.2. 命令示例

```
$ mkdir -p /path/to/output/061.cellcluster
$ singularity exec SAW_v6.1.sif cellCluster \
    -i /path/to/output/051.cellcut/{SN}.cellbin.gef \  ## 必需参数:HDF5 格式的细胞基因表达文件
    -o /path/to/output/061.cellcluster/{SN}.cell.cluster.h5ad  ## 必需参数:细胞聚类 h5ad 文件路径
```

2.11.3. 运行资源

- 内存占用：~5G（1G reads 数据分析参考值）
- 运行时间：~5 分钟（1G reads 数据分析参考值）

2.11.4. 输出文件

```
/path/to/output/061.cellcluster
└── SN.cell.cluster.h5ad   ## 细胞分割聚类产生的 h5ad 文件
```

2.12. saturation

saturation：工具用于计算组织覆盖区域的测序饱和度。

2.12.1. 输入文件

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

2.12.2. 命令示例

```
$ mkdir -p /path/to/output/07.saturation
$ singularity exec SAW_v6.1.sif saturation \
    -i /path/to/output/03.count/{SN}_raw_barcode_gene_exp.txt \   ## 必需参数:count 输出饱和度抽样文件
    --tissue /path/to/output/05.tissuecut/{SN}.tissue.gef \   ## 必需参数:组织分割后 gef 文件
    -o /path/to/output/07.saturation \   ## 必需参数:输出文件目录
    --bcstat /path/to/output/01.mapping/{lane}.CIDMap.stat \   ## 必需参数:CID 比对的统计文件（多对 FASTQ 运行时,此处结果用逗号隔开）
    --summary  /path/to/output/03.count/{SN}.Aligned.sortedByCoord.out.merge.q10.dedup.target.bam.summary.stat   ## 必需参数:注释统计文件
```

2.12.3. 运行资源

- 内存占用：~5G（1G reads 数据分析参考值）
- 运行时间：~5 分钟（1G reads 数据分析参考值）

2.12.4. 输出文件

```
/path/to/output/07.saturation
├── plot_1x1_saturation.png   ##bin1 的测序饱和度分析图
├── plot_200x200_saturation.png   ##bin200 的测序饱和度分析图
└── sequence_saturation.tsv   ## 测序饱和度文件
```

2.13. report

report：工具可帮助用户整合每个步骤生成的分析报告，生成 JSON 格式的结果分析统计报告以及 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、聚类分析结果、以及图像处理信息。mIF 场景下，聚类结果展示的底图增加伪彩 (最多支持 7 种颜色) 效果。

2.13.1. 输入文件

- 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**）
- register 处理过的图像记录文件和图像金字塔文件（**.ipr, .rpi**）
- tissueCut 和 cellCut 输出 GEF 文件、组织覆盖区域统计文件（**.stat**）、统计图 **(.gef, .stat, .png)**
- spatialCluster 和 cellCluter(若已完成）输出聚类 H5AD 文件 **(.h5ad)**
- saturation 输出 bin200 的测序饱和度图 **(.png)**
- 物种、组织和参考信息

2.13.2. 命令示例

场景 1：显微镜染色图像 register 后，同时 cellbin 文件都具备时，运行 report 的输出文件为：

```
$ imageIPR=$(find /path/to/output/04.register -maxdepth 1 -name {SN}*.ipr | head -1)
$ mkdir -p /path/to/output/08.report
$ singularity exec SAW_v6.1.sif report \
    -m /path/to/output/01.mapping/{lane}.CIDMap.stat \  ## 必需参数:CID 比对的统计文件
    -a /path/to/output/01.mapping/{lane}.Log.final.out \  ## 必需参数:STAR 比对统计文件
    -g /path/to/output/03.count/{SN}.Aligned.sortedByCoord.out.merge.q10.dedup.target.bam.summary.stat \  ## 必需参数:带注释 bam 统计文件
    -l /path/to/output/05.tissuecut/tissuecut.stat \  ## 必需参数:组织区域基因统计文件
    -n /path/to/output/05.tissuecut/{SN}.gef \  ## 必需参数:所有 bin 表达 gef 文件
    -d /path/to/output/06.spatialcluster/{SN}.spatial.cluster.h5ad \  ## 必需参数:空间聚类文件
    -t /path/to/output/07.saturation/plot_200x200_saturation.png \  ## 必需参数:bin200 测序饱和度分析图
    -b /path/to/output/05.tissuecut/tissue_fig/scatter_200x200_MID_gene_counts.png \  ## 必需参数:每个 bin (bin 200) 中的 MID 计数和基因数的散点图
    -v /path/to/output/05.tissuecut/tissue_fig/violin_200x200_MID_gene.png \  ## 必需参数:每个 bin (bin 200) 中的 MID 计数和基因数的小提琴图
    -c /path/to/output/05.tissuecut/tissue_fig/statistic_200x200_MID_gene_DNB.png \  ## 必需参数:x 轴含毛边的 MID 数、基因数和 DNB 数的单变量分布图 (bin200)
    --bin20Saturation /path/to/output/05.tissuecut/tissue_fig/scatter_20x20_MID_gene_counts.png \  ## 必需参数:每个 bin (bin 20) 中的 MID 计数和基因数的散点图
    --bin20violin /path/to/output/05.tissuecut/tissue_fig/violin_20x20_MID_gene.png \  ## 必需参数:每个 bin (bin 20) 中的 MID 计数和基因数的小提琴图
    --bin20MIDGeneDNB /path/to/output/05.tissuecut/tissue_fig/statistic_20x20_MID_gene_DNB.png \  ## 必需参数:x 轴含毛边的 MID 数、基因数和 DNB 数的单变量分布图 (bin20)
    --bin50Saturation /path/to/output/05.tissuecut/tissue_fig/scatter_50x50_MID_gene_counts.png \  ## 必需参数:每个 bin (bin 50) 中的 MID 计数和基因数的散点图
```

```
    --bin50violin /path/to/output/05.tissuecut/tissue_fig/violin_50x50_MID_gene.png \  ## 必需参数:每个 bin (bin 50) 中的 MID 计数和基因数的小提琴图
    --bin50MIDGeneDNB /path/to/output/05.tissuecut/tissue_fig/statistic_50x50_MID_gene_DNB.png \  ## 必需参数:x 轴含毛边的 MID 数、基因数和 DNB 数的单变量分布图 (bin50)
    --bin100Saturation /path/to/output/05.tissuecut/tissue_fig/scatter_100x100_MID_gene_counts.png \  ## 必需参数:每个 bin (bin 100) 中的 MID 计数和基因数的散点图
    --bin100violin /path/to/output/05.tissuecut/tissue_fig/violin_100x100_MID_gene.png \  ## 必需参数:每个 bin (bin 100) 中的 MID 计数和基因数的小提琴图
    --bin100MIDGeneDNB /path/to/output/05.tissuecut/tissue_fig/statistic_100x100_MID_gene_DNB.png \  ## 必需参数:x 轴含毛边的 MID 数、基因数和 DNB 数的单变量分布图 (bin100)
    --bin150Saturation /path/to/output/05.tissuecut/tissue_fig/scatter_150x150_MID_gene_counts.png \  ## 必需参数:每个 bin (bin 150) 中的 MID 计数和基因数的散点图
    --bin150violin /path/to/output/05.tissuecut/tissue_fig/violin_150x150_MID_gene.png \  ## 必需参数:每个 bin (bin 150) 中的 MID 计数和基因数的小提琴图
    --bin150MIDGeneDNB /path/to/output/05.tissuecut/tissue_fig/statistic_150x150_MID_gene_DNB.png \  ## 必需参数:x 轴含毛边的 MID 数、基因数和 DNB 数的单变量分布图 (bin150)
    --cellBinGef /path/to/output/051.cellcut/{SN}.cellbin.gef \  ## 可选参数:细胞分割 gef 文件
    --cellCluster /path/to/output/061.cellcluster/{SN}.cell.cluster.h5ad \  ## 可选参数:细胞分割后聚类结果
    -i /path/to/output/04.register/{SN}.rpi \  ## 可选参数:芯片 rpi 文件
    -r standard_version \  ## 必需参数:报告版本
    -s {SN} \  ## 必需参数:芯片号
    --pipelineVersion SAW_v6.1 \  ## 必需参数:流程版本号
    --iprFile ${imageIPR} \  ## 可选参数:图像预处理文件
    --species {species_name} \  ## 必需参数:物种信息
    --tissue {tissue_type} \  ## 必需参数:组织类型
    --reference {reference_index} \  ## 必需参数:参考基因组
    -o /path/to/output/08.report  ## 必需参数:输出文件目录
```

注意：把 {species_name}, {tissue_type}, {reference_index} 替换成真实的信息。
场景 2：当有多对 PE FASTQ 运行 report 时。只需将上述参数中 /path/to/output/ 改成 /path/to/multi_lane_output/，{lane} 改成 {lane*} 即可。
场景 3：使用 rapidRegister 没有细胞分割结果，但与矩阵配准了的图像输出 report。只需将上述参数中“--cellBinGef，--cellCluster”两个参数去掉即可。
场景 4: 无配准图时输出 report。只需将上述参数中“--cellBinGef，--cellCluster，-i，--iprFile”四个参数去掉即可。

2.13.3. 运行资源

- 内存占用：~1G（1G reads 数据分析参考值）
- 运行时间：~1 分钟（1G reads 数据分析参考值）

2.13.4. 输出文件

有 cell bin 时，report 输出文件如下：

```
/path/to/output/08.report
├── scatter_1x1_MID_gene_counts.png  ## 每个细胞的 CID 计数和基因数的散点图
├── SN.report.html  ## 报告 html 文件
└── SN.statistics.json  ## 所有重要数据统计文件
```

```
├── statistic_1x1_cell_area.png   ## 细胞面积沿每个细胞的 x 轴的单变量分布。
├── statistic_1x1_DNB.png   ##DNB 数沿每个细胞 x 轴的单变量分布。
├── statistic_1x1_gene.png  ## 基因类型沿每个细胞 x 轴的单变量分布。
├── statistic_1x1_MID.png   ##MID 计数沿每个细胞 x 轴的单变量分布。
├── violin_1x1_gene.png   ## 小提琴图显示了每个细胞中基因类型的分布。
└── violin_1x1_MID.png   ## 小提琴图显示了每个细胞中重复数据删除的 MID 计数的分布。
```

有 cellbin 时，report 输出文件如下：

```
/path/to/output/08.report
├── SN.report.html   ## 报告 html 文件
└──SN.statistics.json   ## 所有重要数据统计文件
```

第三章
SAW 其他应用工具

3.1 cellCut 其他应用

3.1.1 工具简介

cellCut：是一个可用来处理 GEF 文件的工具。SAW 流程中的这个工具是用来把因表达矩阵的 GEF 格式转变为清晰图表或者补全 GEF 格式。用户也可以使用 C++ 编译好的工具 geftools 或 python 封装的程序包 gefpy 来处理 GEF 文件。

3.1.2 功能示例

功能 1：GEF 转换为 GEM 格式矩阵

```
$ singularity exec SAW_v6.1.sif cellCut view \  ## convert GEF that only contains bin1 geneExp
    -s {SN} \  ## 必需参数:芯片号
    -i /path/to/output/03.count/{SN}.raw.gef \  ## 必需参数:gef 文件
    -o {SN}.raw.gem   ## 可选参数:输出 gem 文件
 $ singularity exec SAW_v6.1.sif cellCut view \  ## convert a whole GEF
    -s {SN} \  ## 必需参数:芯片号
    -i /path/to/output/05.tissuecut/{SN}.gef \  ## 必需参数:gef 文件
    -o {SN}.gem  ## 可选参数:输出 gem 文件
$ singularity exec SAW_v6.1.sif cellCut view \  ## convert tissue GEF that only contains bin1 geneExp
    -s {SN} \   ## 必需参数:芯片号
    -i /path/to/output/05.tissuecut/{SN}.tissue.gef \  ## 必需参数:gef 文件
    -o {SN}.tissue.gem  ## 可选参数:输出 gem 文件
$ singularity exec SAW_v6.1.sif cellCut view \  ## convert cellbin GEF to cellbin GEM
    -s {SN} \  ## 必需参数:芯片号
    -i /path/to/output/051.cellcut/{SN}.cellbin.gef \  ## 必需参数:cellbin 的 gef 文件
    -o {SN}.cellbin.gem \  ## 必需参数:cellbin 的 gem 文件
    -d /path/to/output/03.count/{SN}.raw.gef  ## 可选参数:gef 文件
```

功能 2：GEF 的补全

```
$ singularity exec SAW_v6.1.sif cellCut bgef \  ## complete GEF that only contains bin1 geneExp group to a whole GEF, you may specify the bin size you need using “-b”. Separate multiple bin size with comma
    -i /path/to/output/05.tissuecut/{SN}.tissue.gef \  ## 必需参数:gef 文件
    -o {SN}.tissue.complete.gef \  ## 必需参数:输出 gef 文件
    -b 1,20,50,100 \  ## 必需参数:binsize 大小
    -O Transcriptomics ## 必需参数:组学具体名称
```

功能 3：GEM 转换为 GEF

```
$ singularity exec SAW_v6.1.sif cellCut bgef \  ## convert GEM to GEF in specific bin size. Separate multiple bin sizes with comma
    -i {SN}.gem \   ## 必需参数:输入 gem 文件
    -o {SN}.gef \  ## 必需参数:输出 gef 文件
    -b 1,20,50 \  ## 必需参数:binsize 大小
    -O Transcriptomics  ## 必需参数:组学具体名称
```

3.2 rapidRegister 应用

3.2.1 工具简介

rapidRegister：是一个轻量级的 register 工具，它包含除了细胞分割之外 register 的所有功能。其用法、输入输出文件与 register 均相同。

3.3 checkGTF 应用

3.3.1 工具简介

checkGTF: 是用于检查作为 count 输入文件的 GTF/GEF 的格式是否正确。特别是基因名称长度的限制。

3.3.2 功能示例

```
$ singularity exec SAW_v6.1.sif checkGTF \
    -i /path/to/reference/genes.gtf \ ## 必需参数:基因注释文件 GTF/GFF。
    -o /path/to/reference/genes_new.gtf    ## 必需参数:输出新形式的 GFF/GTF 文件
```

3.4 imageTools 其他应用

3.4.1 工具简介

除了 ipr2img，imageTools 还有三个功能

（1）**img2rpi**（以 rpi 格式写入图像）；

（2）**merge**（将染色图像与组织分割和细胞分割图像融合，以检查分割结果）；

（3）**overlay**（将推断的 track 线模板与拼接的全景图像重叠以检查拼接结果，或将矩阵中的 track 线模板与配准的全景图像重叠，以检查配准结果）。

3.4.2 功能示例

功能 1：img2rpi

```
$ singularity exec SAW_v6.1.sif imageTools img2rpi \
    -i /path/to/output/04.register/<stainType1>_fov_stitched_transformed.tif,/path/to/output/04.register/<stainType2>_fov_stitched_transformed.tif \ ## 必需参数:图像 TIFF 文件(多个图像文件用逗号隔开)
    -g <stainType1>/<ImageType>,<stainType2>/<ImageType>\ ## 必需参数:设置输入保存在 rpi 文件里的命名,芯片号或者图像类型
    -b 1 10 50 100 \ ## 必需参数:binsize 大小
    -o /path/to/output/04.register/fov_stitched_transformed.rpi   ## 必需参数:RPI 输出文件路径,RPI 文件可在 StereoMap 中用来手动配准
```

功能 2：merge（-i 任意最多三个灰度图像（.tif））

```
$ singularity exec SAW_6.1.sif imageTools merge \
    -i /path/to/output/04.register/<stainType>_{SN}_regist.tif,/path/to/output/04.register/{SN}_tissue_cut.tif,/path/to/output/04.register/{SN}_mask.tif \  ## 必需参数:输入 TIFF 文件,多个文件以逗号分割,最多三个
    -o /path/to/output/04.register/merge.tif ## 必需参数:多通道图像合并输出文件路径
```

功能 3：overlay

1. 拼接效果检查：将 IPR 或 TXT 格式的拼接模板与预先配准的全景图像叠加。

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

$ singularity exec SAW_6.1.sif imageTools overlay \
    -i ${preRegImage} \  ## 必需参数:预配准图像文件路径
    -c ${imageIPR} \  ## 必需参数:IPR 文件
    -o /path/to/output/04.register/<stainType>_stitch_check.tif \ ## 必需参数:输出 TIFF 图像路径(拼接检查文件)
    -d stitch \ ## 必需参数:模块名
    -l 60 \ ## 可选参数:Cross limbs length in pixel
    -w 3  ## 可选参数:Cross line thickness in pixel

$ singularity exec SAW_6.1.sif imageTools overlay \
    -i ${preRegImage} \  ## 必需参数:预配准图像文件路径
    -c ${stitchTmplt} \  ## 必需参数:txt 文件
    -o /path/to/output/04.register/<stainType>_stitch_check_tmplt.tif \  ## 必需参数:输出 TIFF 图像路径(拼接检查文件)
    -d stitch \  ## 必需参数:模块名
    -l 60 \  ## 可选参数:Cross limbs length in pixel
    -w 3  ## 可选参数:Cross line thickness in pixel
```

2. 配准效果检查：将 IPR 或 TXT 格式的配准模板与已配准的全景图像叠加。

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

singularity exec SAW_6.1.sif imageTools overlay \
    -i ${regImage} \  ## 必需参数:已配准图像文件路径
    -c ${imageIPR} \  ## 必需参数:IPR 文件
    -o /path/to/output/04.register/register_check.tif \  ## 必需参数:输出 TIFF 图像路径(配准检查文件)
    -d register \  ## 必需参数:模块名
```

```
    -l 60 \  ## 可选参数:Cross limbs length in pixel
    -w 3  ## 可选参数:Cross line thickness in pixel

singularity exec SAW_6.1.sif imageTools overlay \
    -i ${regImage} \  ## 必需参数:已配准图像文件路径
    -c ${regTmplt} \  ## 必需参数:txt 文件
    -o /path/to/output/04.register/register_check_tmplt.tif \  ## 必需参数:输出 TIFF 图像路径(配准检查文件)
    -d register \  ## 必需参数:模块名
    -l 60 \  ## 可选参数:Cross limbs length in pixel
    -w 3  ## 可选参数:Cross line thickness in pixel
```

功能 4：ipr2img

从 IPR 文件输出图像（.tif）。有关此函数的其他用途，请参考手动处理流程的教程。

3.5 manualRegister 应用

3.5.1 工具简介

当自动配准效果不佳时，用户可以使用线下可视化软件 StereoMap 进行手动图像和矩阵的配准。配准后的操作参数记录在 JSON 文件中，可输入进 SAW 的 manualRegister 工具进行图像调整的运算并修改 IPR 中的配准参数记录。用户需要重新运行 imageTools ipr2img 获得修改后的图像。

3.5.2 功能示例

小提示：StereoMap 输出的记录手动配准参数信息 JSON 文件的参数信息与 manualRegister 入参的对应关系：

- **"offsetX"：-o 的 {offsetX}；**
- **"offsetY"：-o 的 {offsetY}；**
- **"flip"：-f 的 {flip}；**
- **"rotate": -r 的 {rotate}；**
- **"extrude_x": -s 的 {extrude_x}；**
- **"extrude_y": -s 的 {extrude_y}；**
- **"isTuned"：-a 的 {isTuned}，注意入参时需要大写首字母。**

场景 1 和场景 2：进行 track 线微调 && 不进行 track 线微调

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

$ singularity exec SAW_v6.1.sif manualRegister \
    -i /path/to/output/04.register \  ## 必需参数:register/rapidRegister 的结果输出目录
    -c ${regIPR} \  ## 必需参数:register/rapidRegister 的 IPR 输出文件
    -v /path/to/output/03.count/{SN}.raw.gef \  ## 必需参数:count 输出 GEF 表达矩阵
```

```
    -o {offsetX} {offsetY} \   ## 可选参数:fov_stitched_transformed.tif 图像中心的 x 和 y 方向偏移量
    -f {flip} \   ## 可选参数:是否沿 y 轴水平翻转图像
    -r {rotate} \   ## 可选参数:从 fov_stitched_transformed.tif 图像中心开始的手动旋转角度
    -s {extrude_x} {extrude_y} \   ## 可选参数:手动缩放 fov_stitched_transformed.tif 图像的 x 和 y 方向
    -a {isTuned} \   ## 可选参数:是否对手动配准的结果进行微调 This need to be a capitalized 'True'
    -p /path/to/output/manualregister   ## 必需参数:输出保存 IPR 结果目录
```

注意：场景 2 不进行 track 线微调的参数只需将上述参数中“-a”去掉即可。

3.6 lasso 应用

3.6.1 工具简介

lasso：根据用户在 StereoMap 中手动圈选出的一个或多个闭合区域进行对应区域的基因表达矩阵的提取。

3.6.2 功能示例

场景 1 及场景 2：square bin lasso && cellbin lasso

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

$ singularity exec SAW_v6.1.sif lasso \
    -i /path/to/output/05.tissuecut/{SN}.gef \   ## 必需参数:gef 文件(如果是 cellbin lasso,此处为 /path/to/output/051.cellcut/{SN}.cellbin.gef)
    -m /upload/path/{taskID}.anno.geojson \   ## 必需参数:stereomap 输出坐标列表文件(*.geojson)
    -o /path/to/output/lasso \   ## 必需参数: 保存 lasso 输出文件的目录
    -s {bin} \   ## 必需参数:bin 的大小
    -n {SN}   ## 必需参数:芯片号
```

注意，如果是场景 2：cellbin lasso，只需将上述参数中“-s”去掉即可。

- Square bin lasso 输出文件如下：

```
/path/to/output/lasso
└── segmentation
    ├── SN.lasso.mask.tif
    └── SN.lasso.label.gem.gz
```

- Cellbin lasso 输出文件如下：

```
/path/to/output/lasso
└── segmentation
    └── SN.lasso.cellbin.gef
```

第四章
SAW 常见 Q&A

4.1. Q: 基因组注释文件 GTF/GFF 有什么格式要求?

4.1.1. 文件格式:

GFF文件 或者 GTF文件,文件后缀名支持 gtf/gtf.gz, gff/gff.gz, gff3/gff3.gz

4.1.2. GTF 文件格式:

注释行以 # 开始

主体部分共 9列,以tab作为分隔符:seqname source feature start end score strand frame attributes

- type:注释信息的类型必须含有gene,transcript 和 exon
- start/end:最大值需小于2^31
- strand:链的正向与负向,分别用加号+和减号-表示。
- 第9列为attributes,格式为tag “value” (标签“值”),不同属性之间以空格相隔 ; 必须要有以下4个
  ①. gene_name value
  ②. gene_id value: 表示转录本在基因组上的基因座的唯一的ID。gene_id与value值用空格分开,如果值为空,则表示没有对应的基因。
  ③. transcript_name value
  ④. transcript_id value: 预测的转录本的唯一ID。transcript_id与value值用空格分开,空表示没有转录本。
- 目前最大有效基因数必须小于2^20,即1048576
- 不可以乱序,即同一个gene的transcript/exons需按顺序排列

4.1.3. GFF 文件格式:

注释行以 # 开始

主体部分共 9列,以tab作为分隔符:seqid source type start end score strand phase attributes

- type:注释信息的类型必须含有gene, mRNA和 exon;
- start/end:最大值需小于2^31;
- strand:“+”表示正链,“−”表示负链,“.”表示不需要指定正负链,“?” 表示未知;
- 第9列为attributes,格式为tag=value (标签=值),不同属性之间以分号相隔;

①. 需要存在ID Name Parent(对gene无需判断Parent);

②. 对于第三列的命名规则请务必仔细研究 ⇒ “树状分级”（不能只列出child行 而没有parent行!）示例如下：

```
##sequence-region ctg123 1 1497228
ctg123 . gene            1000  9000  .  +  .  ID=gene00001;Name=EDEN
ctg123 . TF_binding_site 1000  1012  .  +  .  ID=tfbs00001;Parent=gene00001
ctg123 . mRNA            1050  9000  .  +  .  ID=mRNA00001;Parent=gene00001;Name=EDEN.1
ctg123 . mRNA            1050  9000  .  +  .  ID=mRNA00002;Parent=gene00001;Name=EDEN.2
ctg123 . mRNA            1300  9000  .  +  .  ID=mRNA00003;Parent=gene00001;Name=EDEN.3
ctg123 . exon            1300  1500  .  +  .  ID=exon00001;Parent=mRNA00003
ctg123 . exon            1050  1500  .  +  .  ID=exon00002;Parent=mRNA00001,mRNA00002
ctg123 . exon            3000  3902  .  +  .  ID=exon00003;Parent=mRNA00001,mRNA00003
ctg123 . exon            5000  5500  .  +  .  ID=exon00004;Parent=mRNA00001,mRNA00002,mRNA00003
ctg123 . exon            7000  9000  .  +  .  ID=exon00005;Parent=mRNA00001,mRNA00002,mRNA00003
ctg123 . CDS             1201  1500  .  +  0  ID=cds00001;Parent=mRNA00001;Name=edenprotein.1
ctg123 . CDS             3000  3902  .  +  0  ID=cds00001;Parent=mRNA00001;Name=edenprotein.1
ctg123 . CDS             5000  5500  .  +  0  ID=cds00001;Parent=mRNA00001;Name=edenprotein.1
ctg123 . CDS             7000  7600  .  +  0  ID=cds00001;Parent=mRNA00001;Name=edenprotein.1
ctg123 . CDS             1201  1500  .  +  0  ID=cds00002;Parent=mRNA00002;Name=edenprotein.2
ctg123 . CDS             5000  5500  .  +  0  ID=cds00002;Parent=mRNA00002;Name=edenprotein.2
ctg123 . CDS             7000  7600  .  +  0  ID=cds00002;Parent=mRNA00002;Name=edenprotein.2
ctg123 . CDS             3301  3902  .  +  0  ID=cds00003;Parent=mRNA00003;Name=edenprotein.3
ctg123 . CDS             5000  5500  .  +  1  ID=cds00003;Parent=mRNA00003;Name=edenprotein.3
ctg123 . CDS             7000  7600  .  +  1  ID=cds00003;Parent=mRNA00003;Name=edenprotein.3
ctg123 . CDS             3391  3902  .  +  0  ID=cds00004;Parent=mRNA00003;Name=edenprotein.4
ctg123 . CDS             5000  5500  .  +  1  ID=cds00004;Parent=mRNA00003;Name=edenprotein.4
ctg123 . CDS             7000  7600  .  +  1  ID=cds00004;Parent=mRNA00003;Name=edenprotein.4
```

- 目前最大有效基因数必须小于2^20,即1048576;
- 可以乱序,但仍需满足 gene必须出现在对应mRNA之前,mRNA必须出现在对应的exon之前的规则。

4.1.4. 其他注意事项:

gene/gene_name(基因的名字) 值不含有特殊符号（空格，各类型括号，引号，<>，%等）。支持使用的常见特殊符号有”_“,".";

gene/gene_name(基因的名字) 值长度小于64个字符;

虽然GFF文件现在大部分使用的都是第三版（GFF3），但是文件命名时请命名为.gff ;同理对于GTF文件也请文件命名时采用.gtf。

4.2. Q：读取注释文件时会忽略对应基因的情况有哪些?

gtf格式的属性没有gene_name gene_id transcript_name transcript_id(对gene只需要有gene_name和gene_id);

gff格式的属性没有ID Name Parent(对gene无需判断Parent);

同一个gene下的数据包含多个gene_id,日志打印 "Multiple gene IDs for gene xxx: id1, id2...";

同一个gene下的数据同时包含正反链,日志打印 "Strand disagreement for gene xxx - skipping";

transcript或exon没有transcript_id,日志打印 "Record does not have transcriptID for gene xxx";

同一个gene有多条transcript的transcript_id / ID相同,日志打印 "Transcript appears more than once for xxx";

存在exon的start > end,日志打印 "Exon has 0 or negative extent for xxx";

同一个transcript下的exons之间有overlap,日志打印 "Exons overlap for xxx";

一个gene没有任何transcript,日志打印 "No transcript for gene xxx"。

* 一个contig下出现多条gene有相同的gene_name,合并为一个gene。

4.3. Q：构建 reference 时报错 "Fatal INPUT FILE error, no valid exon lines in the GTF file" 如何处理?

可能是注释GTF/GFF文件和基因组FASTA文件中两者对染色体的命名不完全统一，请注意chromosome name要统一。

4.4. Q：为什么大部分的注释文件中的基因都未被注释上?

大概率是注释文件不规范导致的，请再次参考上述文件格式要求的内容自行排查；

另一种可能性是由于strand正负链符号不规范导致，注释文件中strand值只能是 “+” (forward) 或 “-” (reverse)，请不要和下划线 “_” 搞混。

4.5. Q：是否有工具可以辅助检查注释文件规范?

可以使用SAW sif中的checkGTF工具进行检查，运行方式如下：

```
## export SINGULARITY_BIND="/path/to/input/dir,/path/to/output/dir"
singularity exec SAW_v6.1.sif checkGTF \
    -i <input.gtf/gff> \  ## GTF/GFF file input to be checked
    -o <output.gtf/gff>   ## [optional]. Set to output revised GTF/GFF file. Be aware that this may remove some geneswhich do not meet the requirements and cannot be fixed.
```

可能有部分无法被程序修复的基因注释记录会在输出时被删除，可以在日志中找到记录重新修改GTF文件后再次尝试。

4.6. Q：SAW 中对测序数据做了哪些过滤?

CID过滤：过滤CID无法比对上mask文件的reads；

MID过滤：过滤MID序列中含有N碱基的reads，过滤MID为polyA的reads，过滤含有至少一个以上质量值低于10的碱基的reads；

reads过滤：过滤含有接头和dnb序列的reads。

4.7. Q：基因表达可视化结果异常，没有组织轮廓怎么办?

第一步：检查HTML报告中Valid CID Reads的比例是否正常，不可低于10%，如低于10%请确认测序FASTQ文件和芯片SN对应；

第二步：如Valid CID Reads的比例高于10%，可能存在两种可能性：

- 参考基因组格式异常：Multi-Mapped Reads比例高，Uniquely Mapped Reads比例很低且几乎全部注释失败，需要检查注释使用的GTF/GFF文件是否符合格式要求，是否可以通过检测程序；
- 存在混样的可能性：需要自行排查实验部分。

4.8. Q：如何解析 GEF 格式文件?

Option1:使用C++编译的 geftools:

- **https://github.com/STOmics/geftools**

Option2:使用python包 gefpy:

- **https://pypi.org/project/gefpy/**
- **https://gefpy.readthedocs.io/en/latest/index.html**

Option3:如已安装SAW sif(如v6.1):

- **https://hub.docker.com/repository/docker/stomics/saw**
- **singularity exec SAW_v6.1.sif cellCut**
- **请使用singularity 3.8及以上版本**

```
export HDF5_USE_FILE_LOCKING=FALSE
## gef2gem using geftools
geftools view -i <SN>.gef -o <SN>.gem -s <SN>
# -i input square bin GEF, e.g.SN.raw.gef or SN.gef
# -o output GEM
# -s SN

## gef2gem using gefpy
python
>>> from gefpy.bgef_reader_cy import BgefR
>>> bgef=BgefR(filepath='<SN>.gef',bin_size=200,n_thread=4)
>>> bgef.to_gem('<SN>.bin200.gem')

## gef2gem using SAW sif
## export SINGULARITY_BIND="/path/to/input/dir,/path/to/output/dir"
singularity exec SAW_v6.1.sif cellCut view -i <SN>.gef -o <SN>.gem -s <SN>

## gem2gef
geftools bgef -i <SN>.gem -o <SN>.gef -b 1,20,50 -O Transcriptomics
# -i input square bin GEM
# -o output square bin GEF
# -b bin sizes seqarate by comma, default: 1,10,20,50,100,200,500
# -O omics name
```

4.9 Q：测序在进行 mapping 比对时，FASTQ 太多如何简易处理?

如果需要处理的FASTQ文件太多，更简单的方法是将它们整理成一个FQ_{index}.list文件。FQ_{index}.list的要求是列表文件是将所有索引与拆分mask相同的FASTQs在一起，因为这些文件都是由相同的逻辑拆分的。

示例如下：


```
$ cat SN_SE_fastq_16.list  ## a FASTQ list file gathers all the FASTQs whose index is 16
/path/to/data/lane_1_16.fq.gz
/path/to/data/lane_2_16.fq.gz
```

然后在 {idx}.bcPara文件的in1参数输入FQ_{index}.list文件 的路径。FQ_{index}.list 的{index}和分割mask文件的{index} 务必是相同的。

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

4.10 Q：Valid CID 比例异常应该怎么处理?

Valid CID rate低可以排查的方向有3个：

1. 测序情况。测序质量低会影响比对结果。除了Q30还需要检查call N的情况，可以查看下机报告的base distribution，如果出现N碱基的比例高的情况，就需要考虑是因为测序问题，影响了valid CID rate，最好优先排查。

2. 芯片mask h5文件和FQ不对应。因为mask里记录的CID和对样本测序得到的CID不匹配，导致valid CID rate低。这个情况如果单一出现，一般比例极低，但如果涉及到后一个情况，比例波动比较大，需要酌情判断。

3. 污染/混样。在实验过程中或者建库测序的时候混入了其他样本，因为受到污染，所以影响了valid CID rate。那么可能存在两张芯片，同时可以和这个文库的下机数据比对上。如果混的比较多，可以有明确的组织pattern，如果比例极小，有的情况下会有部分高亮点。”

联系我们

深圳华大生命科学研究院

网址：https://www.stomics.tech

邮箱：services@stomics.tech