

# RNA-seq

---

## 前言

---

转录组测序 (RNA-seq) 研究对象为特定细胞在某一功能状态下所能转录出来的所有RNA的总和, 包括mRNA和非编码RNA。RNA-seq分析内容包括序列比对、转录本拼接、表达定量、差异分析、融合基因检测、可变剪接、RNA编辑和突变检测等。通常的分析不一定需要走全部流程, 按需进行即可。

本文档记录的是本人在学习RNA-seq基本流程的过程记录, 主要包括代码和结果截图, 暂时未包括结果的具体解读, 如需了解该内容请直接上网搜索, 网上相关的教程有很多。

本文档的目标是: 从原始RNA测序数据文件 (通常是fastq文件) 出发, 为每一个样本的RNA的表达进行定量, 即确定每个样本所有基因的表达量。本文档的示例数据是双端测序文件 (单端测序应该大同小异)。

我写本文档的用意一方面是为了记录学习的过程, 熟悉写代码的感觉, 另一方面是提供一个简单易理解的流程供广大读者学习和使用。本文档不仅提供可复现的代码脚本文件, 还提供测试数据, 相信可以让那些想入门转录组测序分析的读者快速上手。由于时间仓促, 本文档暂未包含更多的转录组相关知识和结果解读, 因此建议在学习本文档时结合其他教程学习效果会更好。

本文档以及所有的数据、代码都将公开在我的GitHub账号上, 链接是:

<https://github.com/real793259242/RNA-seq>。

## 运行环境

---

由于RNA-seq上游分析需要占用大量内存 (亲身感受), 因此建议在高性能计算机上完成。另外, RNA-seq涉及许多软件, 软件的安装也是一个问题, 虽说网上教程也有很多, 按部就班也能安装好。幸运的是, 我无意中发现了有人在*BMC Bioinformatics*杂志发表了RNA-seq流程, 叫做 `RASflow`, 并提供了运行该流程的Docker镜像, 这样我只要把作者提供的镜像拉到我电脑里面并进入容器里面就可以直接进行RNA-seq分析而不需要安装软件了, 因为作者已经帮我们装好了! 要特别感谢 `RASflow` 开发者! Docker大法好!

`RASflow` 是一个基于 `Snakemake` 流程语言开发的。因此, `RASflow` 是一个学习转录组测序和 `Snakemake` 语言的有益途径。 `RASflow` Github地址:

<https://github.com/zhxiaokang/RASflow/tree/master>。感兴趣的读者可以访问该链接学习论文和代码。

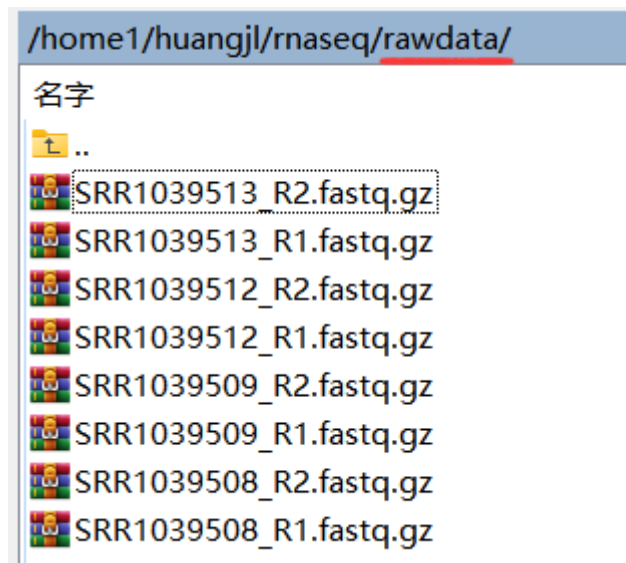
我强烈建议在本地电脑安装docker引擎, 这样就可以使用作者提供的镜像了。假设已经安装好docker, 接着建立容器, 代码是: `docker container run --name=rasflow -it`

zhxiaokang/rasflow。进入容器后激活canda虚拟环境：`conda activate rasflow`，就可以准备进行转录组分析了。

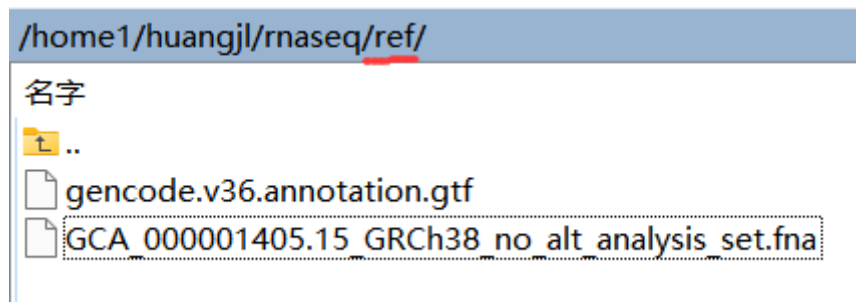
我们先建立一个总文件夹，比如叫做 `rnaseq`，这个就是我们的工作目录。

## 数据获取

可以通过 `sra-tools` 等工具下载NCBI数据库里的原始测序数据fastq文件，下载数据的方式这里暂不赘述。本文档使用的示例数据来自 `RASflow` 项目，再次感谢 `RASflow`。读者可以直接去 `RASflow` 的GitHub地址下载数据。文件是xxx\_R1.fastq.gz、xxx\_R2.fastq.gz格式的双端测序数据。在 `rnaseq` 工作目录下新建一个叫做 `rawdata` 文件夹，把测序数据放在 `rawdata` 里面。如下图：



另外还需下载参考基因组和基因组注释文件。在工作目录 `rnaseq` 下新建 `ref` 文件夹，把下载的参考基因组和注释文件放在 `ref` 里面。如下图：



下载参考基因组和解压代码：

```
wget
ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCA/000/001/405/GCA_000001405.15_GRCh38/seqs_for_alig
nment_pipelines.ucsc_ids/GCA_000001405.15_GRCh38_no_alt_analysis_set.fna.gz && gunzip
GCA_000001405.15_GRCh38_no_alt_analysis_set.fna.gz
```

下载注释文件和解压代码：

```
wget
ftp://ftp.ebi.ac.uk/pub/databases/gencode/Gencode_human/release_36/gencode.v36.annotation.gt
```

```
f.gz && gunzip gencode.v36.annotation.gtf.gz
```

具体下载什么物种的基因组以及什么版本的基因组请按实际需要选择，请参考相关教程。当然 RASflow 项目也提供了参考基因组和注释文件，也可以去那里下载。

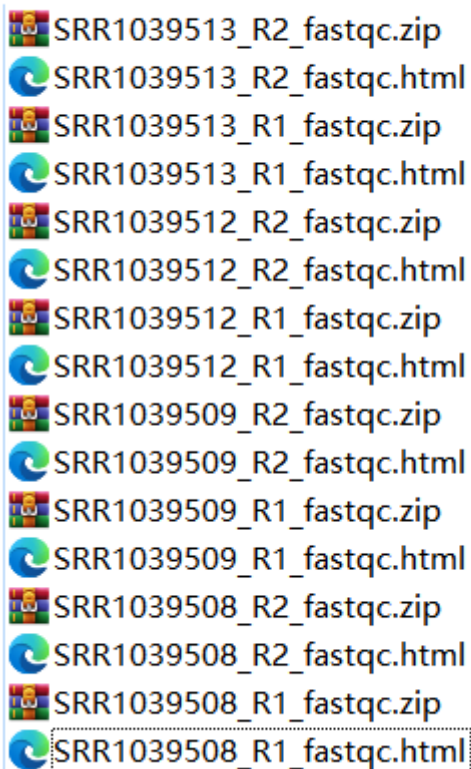
## 质量检测

当获得测序数据后，可以先对它们进行质量检测。使用 `fastqc` 命令进行质量检测。我们先在工作目录 `rnaseq` 下新建 `outputs` 文件夹以存放后续流程的输出结果。接着在 `outputs` 文件夹下新建 `fastqc` 文件夹以存放质量检测的结果。

质量检测的代码示例：`fastqc ./rawdata/*fastq.gz -o ./outputs/fastqc -t 2`，程序运行完成后会输出一堆html文件和zip压缩包，如下图。html是网页版报告，zip是本地报告，下载到本地用浏览器打开就能看到质量检测报告了。

```
/home1/huangjl/rnaseq/outputs/fastqc/
```

名字



SRR1039513\_R2\_fastqc.zip  
SRR1039513\_R2\_fastqc.html  
SRR1039513\_R1\_fastqc.zip  
SRR1039513\_R1\_fastqc.html  
SRR1039512\_R2\_fastqc.zip  
SRR1039512\_R2\_fastqc.html  
SRR1039512\_R1\_fastqc.zip  
SRR1039512\_R1\_fastqc.html  
SRR1039509\_R2\_fastqc.zip  
SRR1039509\_R2\_fastqc.html  
SRR1039509\_R1\_fastqc.zip  
SRR1039509\_R1\_fastqc.html  
SRR1039508\_R2\_fastqc.zip  
SRR1039508\_R2\_fastqc.html  
SRR1039508\_R1\_fastqc.zip  
SRR1039508\_R1\_fastqc.html

关于质量检测报告的解读这里暂时略过，网上教程很多，看以后有没有时间再补上。

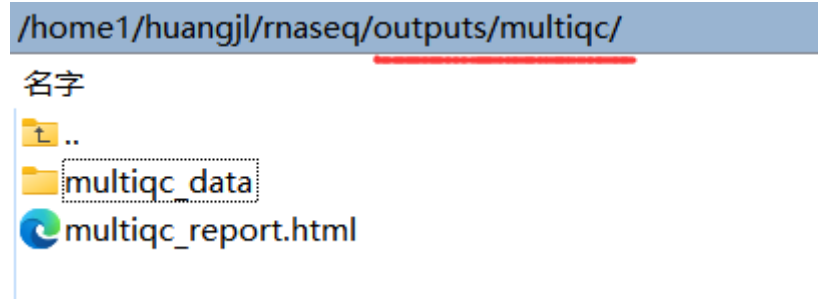
## multiqc整合多个质控结果

`fastqc` 工具针对每一个样本生成一个报告，如果样本比较多的话，逐个打开每份报告检查质量还是比较不方便。`multiqc` 工具能把多个质控结果整合，生成一个HTML网页交互式报告，同时也能导出pdf文件。

我们先在 `outputs` 文件夹下新建 `multiqc` 文件夹以存放 `multiqc` 的输出，代码示例如下：

```
mkdir ./outputs/multiqc
multiqc ./outputs/fastqc -o ./outputs/multiqc
```

如下图：



同样地，关于 `multiqc` 的质量检测报告的解读这里暂时略过，网上教程很多，看以后有没有时间再补上。

## 过滤低质量序列

对数据进行彻底的质量控制并采取适当的步骤来去除问题对于几乎所有测序应用的分析至关重要。这里使用 `trim_galore` 工具对测序数据进行质量控制，过滤低质量序列。我们先在 `outputs` 文件夹下新建 `fastqc_trimmed` 文件夹以存放 `trim_galore` 的输出，示例代码如下：

```
mkdir ./outputs/fastqc_trimmed

# 提取所有样本名称
samples=$(ls rawdata/ | grep '_R1.fastq' | awk -F '_R1.fastq' '{print $1}')
# echo查看样本名称
echo $samples

for sample in $samples
do
    trim_galore --fastqc -j 4 --paired --basename $sample -o ./outputs/fastqc_trimmed
    "rawdata/"$sample"_R1.fastq.gz" "rawdata/"$sample"_R2.fastq.gz"
done
```

以上代码的作用是对低质量序列进行过滤后生成fastq文件（后缀名fq.gz）并对它们运行 `fastqc`（参数 `--fastqc` 的作用）。如下图：

/home1/huangjl/rnaseq/outputs/fastqc\_trimmed/

名字



SRR1039513\_R2\_val\_2\_fastqc.zip  
SRR1039513\_R2\_val\_2\_fastqc.html  
SRR1039513\_R1\_val\_1\_fastqc.zip  
SRR1039513\_R1\_val\_1\_fastqc.html  
SRR1039513\_R2\_val\_2.fq.gz  
SRR1039513\_R2.fastq.gz\_trimming\_report.txt  
SRR1039513\_R1\_val\_1.fq.gz  
SRR1039513\_R1.fastq.gz\_trimming\_report.txt  
SRR1039512\_R2\_val\_2\_fastqc.zip  
SRR1039512\_R2\_val\_2\_fastqc.html  
SRR1039512\_R1\_val\_1\_fastqc.zip  
SRR1039512\_R1\_val\_1\_fastqc.html  
SRR1039512\_R2\_val\_2.fq.gz  
SRR1039512\_R2.fastq.gz\_trimming\_report.txt  
SRR1039512\_R1\_val\_1.fq.gz  
SRR1039512\_R1.fastq.gz\_trimming\_report.txt  
SRR1039509\_R2\_val\_2\_fastqc.zip

## 建立索引

接下来使用 `hisat2` 工具将质控后的RNA片段 (reads) 比对到参考基因组，从而确定这些RNA片段在基因组上的精确位置，进一步可以用于基因表达量定量、剪接位点的检测等多种RNA-seq分析任务。

`hisat2` 需要一个index索引才能进行比对。`hisat2` 提供了一些index，但很少，只有人类、小鼠等基因组，可以在<ftp://ftp.ccb.jhu.edu/pub/infphilo/hisat2/data>中进行下载。也可以自己建立索引，使用 `hisat2-build` 工具以及前面下载好的参考基因组和注释文件来建立。我们先在 `outputs` 文件夹下新建 `hisat2_index` 文件夹以存放索引。示例代码如下：

```
mkdir outputs/hisat2_index
hisat2_extract_splice_sites.py ref/gencode.v36.annotation.gtf >
outputs/hisat2_index/gencode.v36.annotation.ss
hisat2_extract_exons.py ref/gencode.v36.annotation.gtf >
outputs/hisat2_index/gencode.v36.annotation.exon
hisat2-build -p 4 --ss outputs/hisat2_index/gencode.v36.annotation.ss --exon
outputs/hisat2_index/gencode.v36.annotation.exon
ref/GCA_000001405.15_GRCh38_no_alt_analysis_set.fna
outputs/hisat2_index/gencode.v36.annotation_tran
```












# `extract_splice_sites.py`脚本从`gtf`注释文件提取剪接位点信息

# extract\_exons.py脚本从gtf注释文件提取外显子信息

建立索引的过程比较耗时且内存占用比较大。下图是运行过程中的截图：

```
root@9050559e88db: /home/rnaseq
MADE NEW EDGES: 146
SORTED NEW EDGES: 431
RE-SORTED NODES: 474
PROCESS EDGES: 50
REMOVE Y: 8
SORT, Make index: 355
TOTAL: 1593
Allocating ftab, absorbFtab
Entering GFM loop
Exited GFM loop
fchr[A]: 0
fchr[C]: 868294576
fchr[G]: 1468670996
fchr[T]: 2071185218
fchr[$]: 2941947422
Exiting GFM::buildToDisk()
Returning from initFromVector
Wrote 1814662170 bytes to primary GFM file: outputs/hisat2_index/gencode.v36.annotation_tran.1.ht2
Wrote 735391944 bytes to secondary GFM file: outputs/hisat2_index/gencode.v36.annotation_tran.2.ht2
Re-opening in1 and in2 as input streams
Returning from GFM constructor
Returning from initFromVector
Wrote 1747474765 bytes to primary GFM file: outputs/hisat2_index/gencode.v36.annotation_tran.5.ht2
Wrote 747553088 bytes to secondary GFM file: outputs/hisat2_index/gencode.v36.annotation_tran.6.ht2
Re-opening in5 and in6 as input streams
Returning from HierBwt constructor
Headers:
  len: 2934876451
  gbwtLen: 2941947423
  nodes: 2941567751
  sz: 1467438226
  gbwtSz: 1470973712
  lineRate: 8
  offRate: 4
  offMask: 0xffffffff0
  ftabChars: 10
  eftabLen: 0
  eftabSz: 0
```

hisat2-build 运行完成之后结果会存放在 hisat2\_index 文件夹下，进去之后可以看到8个以 ht2 为拓展名的文件。这些文件都比较大，因此在建立索引时要保证充足的硬盘空间。如下图：

/home1/huangjl/rnaseq/outputs/hisat2_index/*.*	
名字	大小
	
 gencode.v36.annotation_tran.6.ht2	730,033 KB
 gencode.v36.annotation_tran.5.ht2	1,706,519 KB
 gencode.v36.annotation_tran.2.ht2	718,157 KB
 gencode.v36.annotation_tran.1.ht2	1,772,132 KB
 gencode.v36.annotation_tran.8.ht2	2,719 KB
 gencode.v36.annotation_tran.7.ht2	13,896 KB
 gencode.v36.annotation_tran.4.ht2	716,523 KB
 gencode.v36.annotation_tran.3.ht2	11 KB
 gencode.v36.annotation.exon	8,217 KB
 gencode.v36.annotation.ss	9,774 KB

## 比对 (hisat2)



索引建立完之后就可以进行比对了。我们先在 `outputs` 文件夹下新建 `hisat2_alignment` 文件夹以存放 `hisat2` 输出结果（sam文件）。示例代码如下：

```
mkdir ./outputs/hisat2_alignment






for sample in $samples
do
    hisat2 -p 4 --dta -x outputs/hisat2_index/gencode.v36.annotation_tran -1
    "outputs/fastqc_trimmed/"$sample"_R1_val_1.fq.gz" -2
    "outputs/fastqc_trimmed/"$sample"_R2_val_2.fq.gz" -S
    "outputs/hisat2_alignment/"$sample".sam"
done
```

运行完成后会输出sam文件，还会输出一段信息，如下图。这些信息的具体解释暂时略过，不过可以看到每个样本最后的overall alignment rate信息，即比对成功率。在RNA-seq分析中，比对成功率是一个重要的质量控制指标。可以看到所有样本的比对成功率都在98%以上，说明所有样本的reads序列都能够成功地比对到基因组上，这表示RNA-seq实验和测序质量都相对较好。

```
root@bd72eea30ace: /home/rnaseq
98.24% overall alignment rate
48077 reads; of these:
  48077 (100.00%) were paired; of these:
    1694 (3.52%) aligned concordantly 0 times
    38147 (79.35%) aligned concordantly exactly 1 time
    8236 (17.13%) aligned concordantly >1 times
  ----
    1694 pairs aligned concordantly 0 times; of these:
      222 (13.11%) aligned discordantly 1 time
  ----
    1472 pairs aligned 0 times concordantly or discordantly; of these:
      2944 mates make up the pairs; of these:
        1535 (52.14%) aligned 0 times
        870 (29.55%) aligned exactly 1 time
        539 (18.31%) aligned >1 times
98.40% overall alignment rate
49413 reads; of these:
  49413 (100.00%) were paired; of these:
    1796 (3.63%) aligned concordantly 0 times
    37662 (76.22%) aligned concordantly exactly 1 time
    9955 (20.15%) aligned concordantly >1 times
  ----
    1796 pairs aligned concordantly 0 times; of these:
      628 (34.97%) aligned discordantly 1 time
  ----
    1168 pairs aligned 0 times concordantly or discordantly; of these:
      2336 mates make up the pairs; of these:
        1060 (45.38%) aligned 0 times
        768 (32.88%) aligned exactly 1 time
        508 (21.75%) aligned >1 times
98.93% overall alignment rate
46560 reads; of these:
  46560 (100.00%) were paired; of these:
    1932 (4.15%) aligned concordantly 0 times
    37524 (80.59%) aligned concordantly exactly 1 time
    7104 (15.26%) aligned concordantly >1 times
  ----
    1932 pairs aligned concordantly 0 times; of these:
      476 (24.64%) aligned discordantly 1 time
  ----
    1456 pairs aligned 0 times concordantly or discordantly; of these:
      2912 mates make up the pairs; of these:
        1447 (49.69%) aligned 0 times
        935 (32.11%) aligned exactly 1 time
        530 (18.20%) aligned >1 times
98.45% overall alignment rate
```

sam文件存放在 hisat2\_alignment 文件夹下，如下图：



/home1/huangjl/rnaseq/outputs/hisat2_alignment/	
名字	大小
	
 SRR1039513.sam	27,271 KB
 SRR1039512.sam	29,987 KB
 SRR1039509.sam	28,635 KB
 SRR1039508.sam	29,083 KB










## samtools排序压缩

把sam文件转换成bam文件。使用 `samtools view` 将sam格式文件转换为bam格式，使用 `samtools sort` 对bam文件进行排序。我们先在 `outputs` 文件夹下新建 `samtools_bam` 文件夹以存放 `samtools` 输出结果（bam文件）。示例代码如下：

```
mkdir ./outputs/samtools_bam

for sample in $samples
do
    samtools view -bS "outputs/hisat2_alignment/"$sample".sam" >
"outputs/samtools_bam/"$sample".bam"
    samtools sort "outputs/samtools_bam/"$sample".bam" -o
"outputs/samtools_bam/"$sample".sort.bam"
done
```

bam文件存放在 `samtools_bam` 文件夹下，如下图：

/home1/huangjl/rnaseq/outputs/samtools_bam/	
名字	大小
	
 SRR1039513.sort.bam	5,282 KB
 SRR1039513.bam	6,264 KB
 SRR1039512.sort.bam	4,966 KB
 SRR1039512.bam	5,953 KB
 SRR1039509.sort.bam	4,824 KB
 SRR1039509.bam	5,928 KB
 SRR1039508.sort.bam	4,915 KB
 SRR1039508.bam	5,995 KB









# featureCounts生成基因计数表

有了bam文件后，接下来使用 featureCounts 工具对基因表达进行定量。我们先在 outputs 文件夹下新建 featureCounts 文件夹以存放 featureCounts 输出结果。示例代码如下：

```
mkdir ./outputs/featureCounts

for sample in $samples
do
    featureCounts -p -T 4 -t exon -g gene_id -a ref/gencode.v36.annotation.gtf -o
    "outputs/featureCounts/"$sample"_count.tsv" "outputs/samtools_bam/"$sample".sort.bam"
done
```

featureCounts 为每一个样本生成两个文件：xxx\_count.tsv和xxx\_count.tsv.summary。  
xxx\_count.tsv.summary文件是计数统计情况，xxx\_count.tsv文件是基因的具体信息，如下图：

/home1/huangjl/rnaseq/outputs/featureCounts/	
名字	大小
 SRR1039513_count.tsv.summary	1 KB
 SRR1039513_count.tsv	37,598 KB
 SRR1039512_count.tsv.summary	1 KB
 SRR1039512_count.tsv	37,598 KB
 SRR1039509_count.tsv.summary	1 KB
 SRR1039509_count.tsv	37,598 KB
 SRR1039508_count.tsv.summary	1 KB
 SRR1039508_count.tsv	37,598 KB

下图是xxx\_count.tsv的部分文件内容，第一列是基因ID，最后一列是每个基因对应的比对到基因组的reads数量，也可以认为是表达量。

```
(rasflow) root@bd72eea30ace:/home/rnaseq/outputs/featureCounts# tail -20 SRR1039508_count.tsv
ENSG00000210154.1    chrM    7518    7585    +        68      1
ENSG00000198712.1    chrM    7586    8269    +        684     2816
ENSG00000210156.1    chrM    8295    8364    +        70       0
ENSG00000228253.1    chrM    8366    8572    +        207     567
ENSG00000198899.2    chrM    8527    9207    +        681    3363
ENSG00000198938.2    chrM    9207    9990    +        784    3382
ENSG00000210164.1    chrM    9991    10058   +         68       0
ENSG00000198840.2    chrM    10059   10404   +        346       0
ENSG00000210174.1    chrM    10405   10469   +         65       0
ENSG00000212907.2    chrM    10470   10766   +        297       0
ENSG00000198886.2    chrM    10760   12137   +       1378       0
ENSG00000210176.1    chrM    12138   12206   +         69       0
ENSG00000210184.1    chrM    12207   12265   +         59       0
ENSG00000210191.1    chrM    12266   12336   +         71       0
ENSG00000198786.2    chrM    12337   14148   +       1812       0
ENSG00000198695.2    chrM    14149   14673   -        525       0
ENSG00000210194.1    chrM    14674   14742   -         69       0
ENSG00000198727.2    chrM    14747   15887   +       1141       0
ENSG00000210195.2    chrM    15888   15953   +         66       0
ENSG00000210196.2    chrM    15956   16023   -         68       0
```

至此，我们就完成了从原始测序数据到每个样本的RNA表达定量的一系列过程。

## RNA-seq (Snakemake版本)

---

上述的RNA-seq流程都是用shell脚本来完成每一个步骤，把这些shell脚本串在一起形成一个总的脚本文件即完成了对生信流程的实现。然而这种实现方式有许多局限性，比如无法监控任务运行状态、无法查看任务日志、无法并行、无法断点运行、修改麻烦等问题。

Snakemake 能很好解决上述问题。Snakemake 是一个基于 Python 的工作流程管理系统，非常适合新手用户以及熟悉 Python 的用户，因为它集成了 Python 语言简单易读、逻辑清晰、便于维护的特点，同时支持 Python 语法。Snakemake 支持多线程、断点运行，支持shell命令，还可以和 Python 库结合使用。

本文档用 Snakemake 语言对上一节的RNA-seq流程进行编写，一方面是熟悉和学习 Snakemake 语言，另一方面是为了以后可能的项目做好准备，如果以后有类似的任务需求，只需一行代码即可自动完成一系列的流程步骤，十分方便快捷。

本人已经编写好 Snakemake 流程脚本，命名为 Snakefile.py，这个脚本我也会共享到GitHub上。下面介绍如何使用本流程。

## 准备工作

---

和上一节一样，我们先建立一个总文件夹，比如叫做 project4\_RNAseq，这个就是我们的工作目录。

把写好的 Snakefile.py 放在 project4\_RNAseq 文件夹里面。

在 project4\_RNAseq 文件夹下新建 data/fastq\_pair/ 文件夹，并把原始测序数据放在 fastq\_pair 文件夹里面。

在 project4\_RNAseq 文件夹下新建 data/ref/genome/ 文件夹，并把参考基因组文件放在 genome 文件夹里面；在 project4\_RNAseq 文件夹下新建 data/ref/annotation/ 文件夹，并把注释文件放在 annotation 文件夹里面。

在 project4\_RNAseq 文件夹新建 configs 文件夹，在 configs 文件夹里新建一个配置文件命名为 config\_main.yaml。在 Snakemake 中允许从一个yaml文件中读取配置参数，这样可以使工作流的管理更加灵活和高效。在本文档，yaml配置文件主要是定义新建文件夹的名称、下载链接、线程数等配置，如下图：

```

1 # This configure yaml file is used to define the parameters for the whole RNA-seq pipeline.
2
3
4 ## Set directory names.
5 fastq_dir: fastq_pair # Put raw fastq data here.
6 reference_dir: ref
7 reference_genome_dir: genome
8 reference_transcriptome_dir: transcriptome
9 reference_annotation_dir: annotation
10 log_dir: logs
11 output_dir: outputs
12 output_fastqc_dir: fastqc
13 output_index_dir: hisat2_index
14 output_alignment_dir: hisat2_alignment
15 output_sam2bam_dir: sam2bam
16 output_featureCounts_dir: featureCounts
17
18
19 ## Reference url.
20 reference_genome_download_url: ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCA/000/001/405/GCA_000001405.15_GH
21 reference_annotation_download_url: ftp://ftp.ebi.ac.uk/pub/databases/gencode/Gencode_human/release_36/gen
22
23
24
25
26 ## Which mapping reference do you want to use? Genome or transcriptome?
27 REFERENCE: genome # "genome" or "transcriptome" (code is not implemented when 'transcriptome'.)
28
29
30 ## Do you need to do trimming?
31 TRIMMED: yes # "yes" or "no". If yes, outputs/fastqc_trimmed/ will be created.
32
33
34 ## Number of threads
35 num_threads: 4

```

## 运行环境

本流程运行环境和上一节一样，也是在 `RASflow` 作者提供的docker容器下运行的，容器已经安装好 `Snakemake` 软件了。同样也是建议在服务器下运行流程。

## 使用方法

当准备工作做完后，运行 `Snakemake` 流程就很简单了。在工作目录 `project4_RNAseq` 下，命令行运行：`snakemake -r -s Snakefile.py`，`Snakemake` 就会自动执行上一节的RNA-seq流程了。

```

(rasflow) root@bd72eea30ace:/home/snakemake/project4_RNAseq# snakemake -r -s Snakefile.py
Building DAG of jobs...
Nothing to be done.
RNA-seq workflow start.
Complete log: /home/snakemake/project4_RNAseq/.snakemake/log/2024-05-08T124157.609580.snakemake.log
RNA-seq workflow finished, no error.
(rasflow) root@bd72eea30ace:/home/snakemake/project4_RNAseq#

```

程序会在工作目录新建 `outputs` 文件夹，并把流程中的每一个步骤的输出放在 `outputs` 文件夹下，如下图所示：

```
/home1/huangjl/snakemake/project4_RNAseq/outputs/
```

名字



fastqc

fastqc\_trimmed

featureCounts

hisat2\_alignment

hisat2\_index

sam2bam

Snakemake 程序目前没出现报错。至此先告一段落。

## 结语

转录组学是生信分析的基础，学习转录组上游数据分析对于生信初学者来说是一个不错的入门方式。我学习RNA-seq并记录学习过程除了巩固自身的生信分析能力，还想为广大对生信感兴趣但不知从何下手的人员提供一个简单且可复现的项目。

由于时间仓促，很多转录组相关的内容本文档暂未包含，比如结果的解读、程序参数的含义等。我本人不是生信科班出身，对生物学方面的认识肯定不如网上教程那么的准确和全面，但看未来有没有时间我也可以把我对这方面的理解补充上去。因此目前的话，读者只看本文档的话肯定不够，结合网上别人的教程使用会更好。不过本文档提供了较为完整的、可复现的代码，我个人认为还是挺少见的。

另外，本文档的RNA-seq流程也不是完整的流程，像后续表达矩阵的生成、差异分析、富集分析等内容暂未包含，不过个人认为这些已不是什么难事了，RNA-seq中最难的部分像参考基因组索引建立、比对、表达定量都包含了，如果读者有一些编程基础，表达矩阵也就很容易获得，有了表达矩阵后续的数据分析其实也都不难了。

我会将本文档涉及的所有代码（包括shell脚本和Snakemake脚本）都共享在我的GitHub主页上，地址是：<https://github.com/real793259242/RNA-seq>。如果对本项目有任何问题（文档内容、代码运行等）、批评、建议等，请不要犹豫通过邮件联系我：[793259242@qq.com](mailto:793259242@qq.com)。十分感谢。

## 参考

1. trim\_galore参考链接：

[https://blog.csdn.net/weixin\\_44616693/article/details/131378368](https://blog.csdn.net/weixin_44616693/article/details/131378368)

2. [http://360doc.com/content/23/0927/14/1098188476\\_1098188476.shtml](http://360doc.com/content/23/0927/14/1098188476_1098188476.shtml)

3. RASflow项目地址：<https://github.com/zhxiaokang/RASflow/tree/master>

4. RNA-seq英文教程：<https://docs.tinybio.cloud/docs/rna-seq-tutorial-from-fastq-to-figures>