Snakemake
- pip安装
- 示例使用:
- 规则的执行顺序
- 可视化流程图
- 参数

Snakemake

官方文档：https://snakemake.readthedocs.io/en/stable/

文章：Snakemake—a scalable bioinformatics workflow engine | Bioinformatics | Oxford Academic (oup.com)

pip安装

```
pip install snakemake
```

Snakemake是一个用于管理计算生物学工作流程的基于Python语言的开源软件。它允许用户通过简单、清晰的脚本定义数据处理过程中不同任务之间的依赖关系，并自动识别其中的并行计算机会，从而实现高效地计算资源利用。

在使用Snakemake时，用户可以通过编写一个包含若干规则（rule）的Snakefile文件来描述数据分析的流程。每个规则通常定义了一个目标文件及其生成所需的输入文件、命令行参数、程序路径等必要信息，同时指定了如何生成这个目标文件的shell 或脚本命令。在执行Snakefile时，Snakemake会根据规则之间的相关性和具体执行情况确定需要执行哪些任务，以后台进程的形式完成数据分析流程。

除了方便地管理复杂的分析流程外，Snakemake还提供了许多便捷的功能，如支持容器化操作、错误恢复机制、远程资源调度等，也可以与其他软件比如conda、DRMAA集成使用。因此，它已经被广泛应用于许多生物信息学领域，包括基因表达、全基因组测序、重测序、转录组学等研究项目

Snakemake 工作流是通过在 Snakefile 中指定命令来定义的。 命令通过指定如何从**输入文件集**创建**输出文件集**，**将工作流分解为小步骤**（例如，单个工具的应用）。Snakemake通过匹配文件名自动确定命令之间的**依赖关系。**

示例使用:

将input.txt中所有小写字母转换为大写字母

```
root@bb25a868318a:/home# cat input.txt
bcdefghijklmnopqrstuvwxyz
root@bb25a868318a:/home# snakemake -s zhuan.py -j 1
Building DAG of jobs...
Using shell: /usr/bin/bash
Provided cores: 1 (use --cores to define parallelism)
Rules claiming more threads will be scaled down.
Job stats:
job      count    min threads    max threads
-----  -------  -------------  -------------
zhuan        1              1              1
total        1              1              1

Select jobs to execute...

[Wed May 31 07:47:37 2023]
rule zhuan:
    input: input.txt
    output: output.txt
    jobid: 0
    reason: Missing output files: output.txt
    resources: tmpdir=/tmp

[Wed May 31 07:47:37 2023]
Finished job 0.
1 of 1 steps (100%) done
Complete log: .snakemake/log/2023-05-31T074737.662993.snakemake.log
root@bb25a868318a:/home# cat output.txt
BCDEFGHIJKLMNOPQRSTUVWXYZ
```

snakefile:

```
rule zhuan:
    input:"input.txt"
    output:"output.txt"
    shell:
        "tr 'a-z' 'A-Z' < {input}> {output}"
```

snakemake还可以调用外部的脚本，包括python、R、shell脚本:

```
rule example4:
  input: a = "data.txt"
  output: b = "result.txt"
  script: "script.R"
```

规则的执行顺序

最基本的原则是通过输入和输出文件的依赖关系来判断规则之间的执行顺序。如果一个规则的输出文件是另一个规则的输入文件，那么就要先执行前者，再执行后者。例如，规则A的输出文件是output_file，规则B的输入文件是它。那么就要先执行规则A，生成output_file，然后再执行规则B。

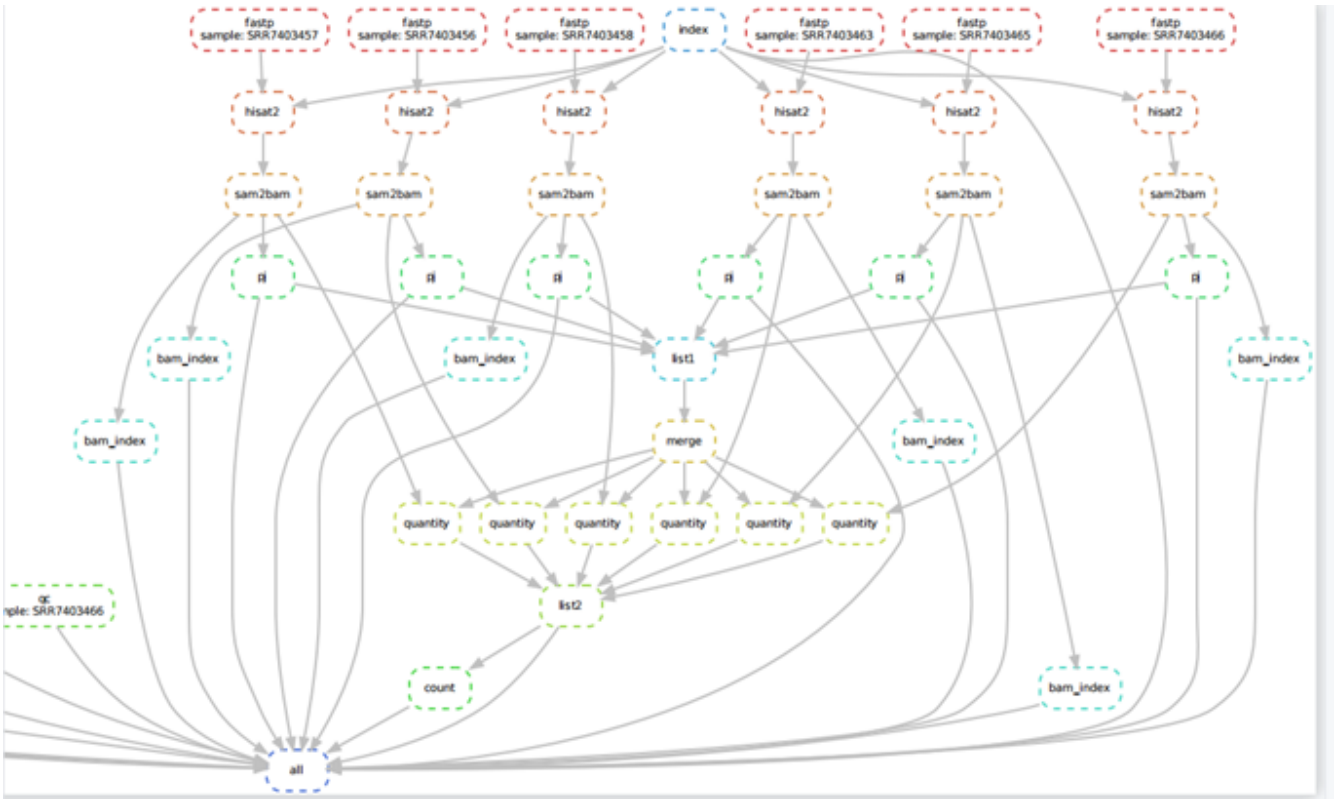
Snakemake会根据每个规则的输入和输出来确定它们的执行顺序和依赖关系。默认情况下，Snakemake会执行第一个规则，然后根据该规则的输入来决定接下来执行哪个规则。为了调用所有规则，我们需要在工作流文件的开头编写“all”规则，它的输入文件是整个流程最终的输出文件。这样，Snakemake就可以根据输入和输出文件的对应关系，层层推导出所有需要执行的规则以及规则之间的顺序。

如:

```
root@bb25a868318a:/home# cat all_cs.py
SAMPLES=["SRR7403456","SRR7403457","SRR7403458","SRR7403463","SRR7403465","SRR7403466"]

rule all:
    input:
        expand("/home/QC/{sample}_1_fastqc.html",sample=SAMPLES),
        expand("/home/QC/{sample}_2_fastqc.html",sample=SAMPLES),
        "/home/QC/multiqc.html",
        expand("/home/reference/genome_tran.{i}.ht2",i=range(1,8)),
        expand("/home/mapping/{sample}.bam.bai",sample=SAMPLES),
        expand("/home/mapping/{sample}.gtf",sample=SAMPLES),
        "/home/quantity/transcript_fpkm_matrix.csv",
        "/home/quantity/list.txt"

rule qc:
    input:
        "/home/fastq/{sample}/{sample}_1.fastq.gz",
        "/home/fastq/{sample}/{sample}_2.fastq.gz"
    output:
        "/home/QC/{sample}_1_fastqc.html",
        "/home/QC/{sample}_1_fastqc.zip",
        "/home/QC/{sample}_2_fastqc.html",
        "/home/QC/{sample}_2_fastqc.zip"
    log:
        "/home/QC/qc_{sample}.log"
    shell:
        "fastqc -t 4 -o /home/QC {input[0]} {input[1]} > {log} 2>&1"

rule multiqc:
    input:
        expand("/home/QC/{sample}_1_fastqc.html",sample=SAMPLES),
        expand("/home/QC/{sample}_2_fastqc.html",sample=SAMPLES)
    output:
```

可视化流程图

```
snakemake -s all_cs.py --dag | dot -Tpdf > dag.pdf

# apt-get install -y graphviz (如果没有dot命令)
```



参数

Snakemake的参数非常多，常用的有以下几个：

- -p：打印运行的shell命令。
- -n：只展示需要完成的步骤，不运行。
- -F：强制运行所有步骤。
- -j：并行运行多个任务。
- -s : 指定snakefile文件