Biological workflows

- Snakemake Tutorial

刘博

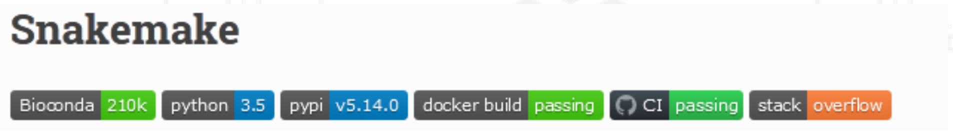
2020.04

Snakemake Tutorial

1 **基础规则介绍**

2 进阶使用

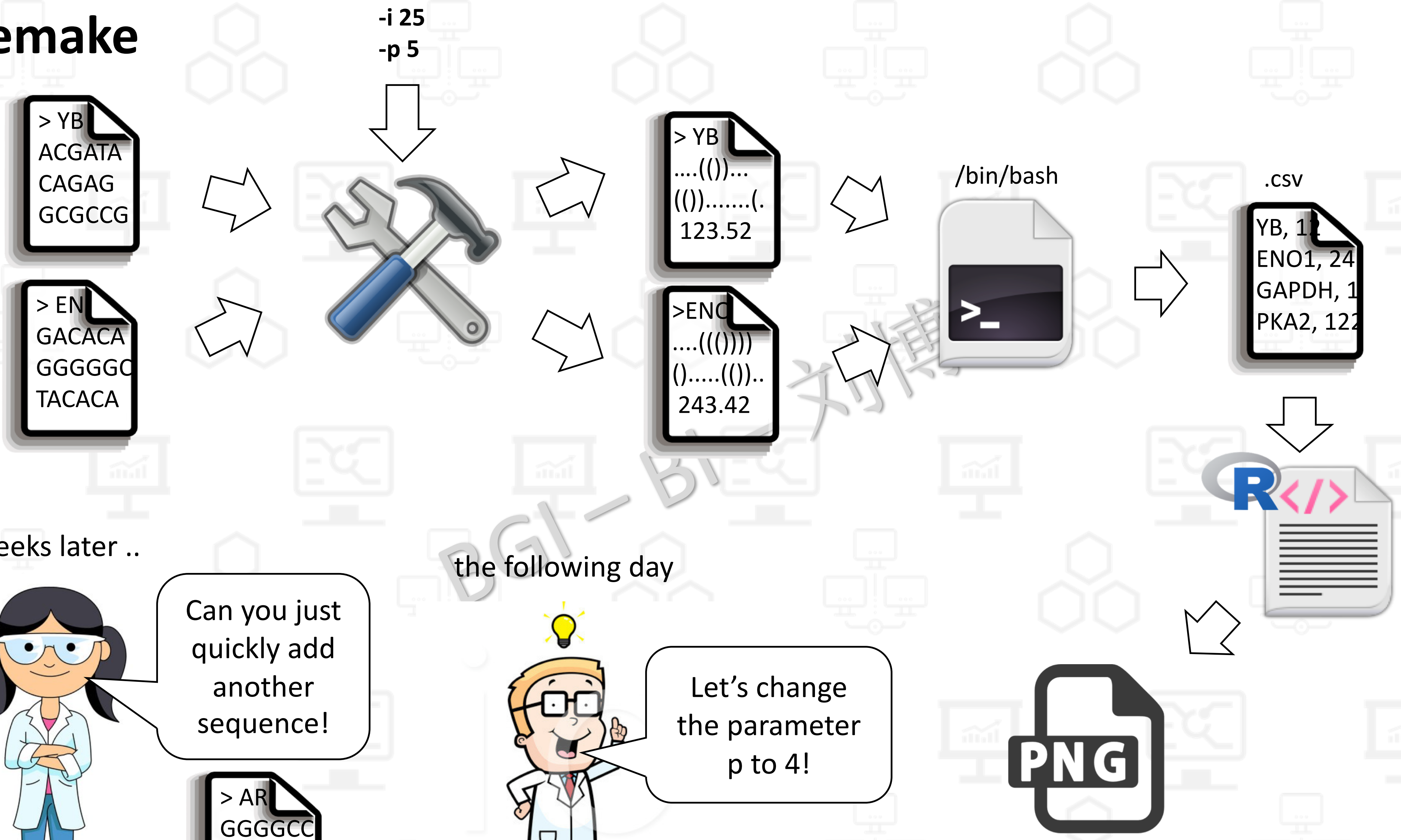
3 扩展功能


Snakemake

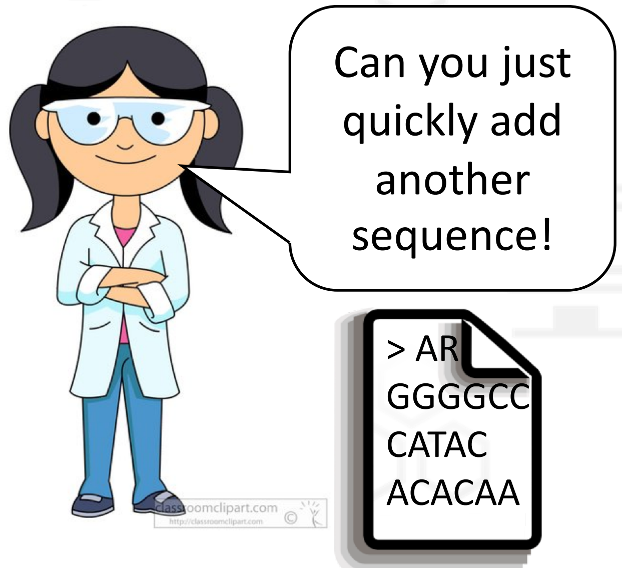
Bioconda 210k | python 3.5 | pypi v5.14.0 | docker build passing | CI passing | stack overflow

https://snakemake.readthedocs.io/

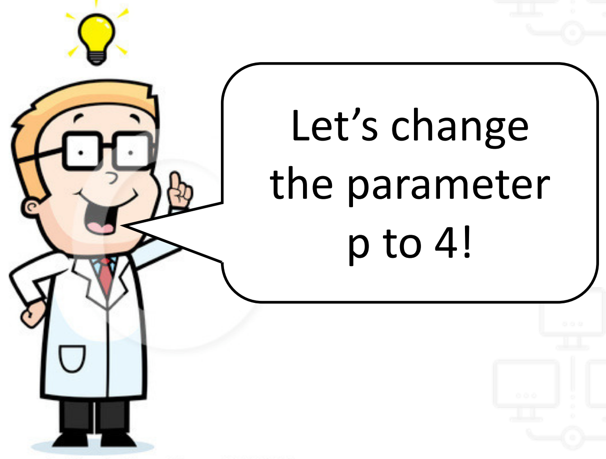
Snakemake



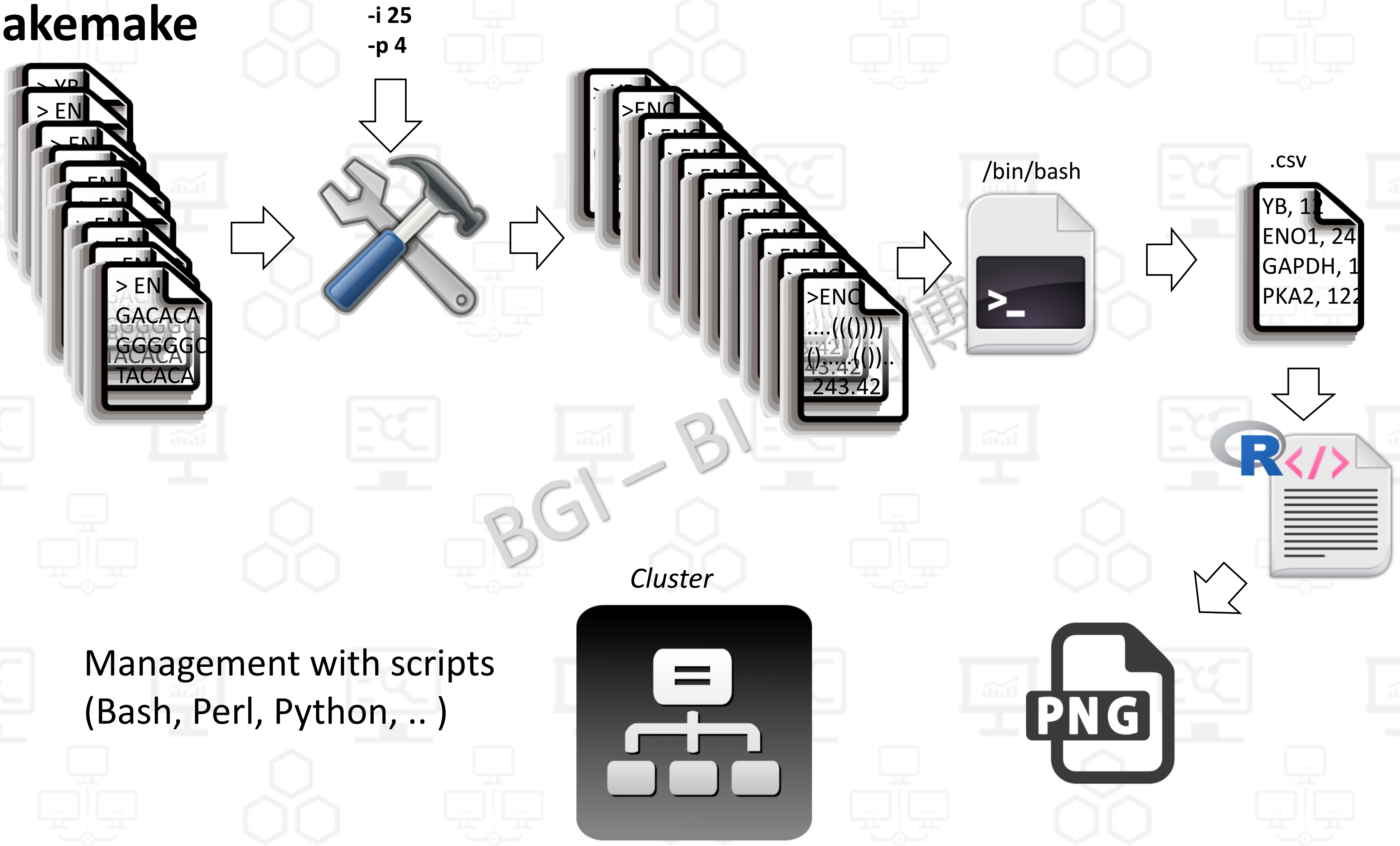
2 weeks later ..



the following day



Snakemake

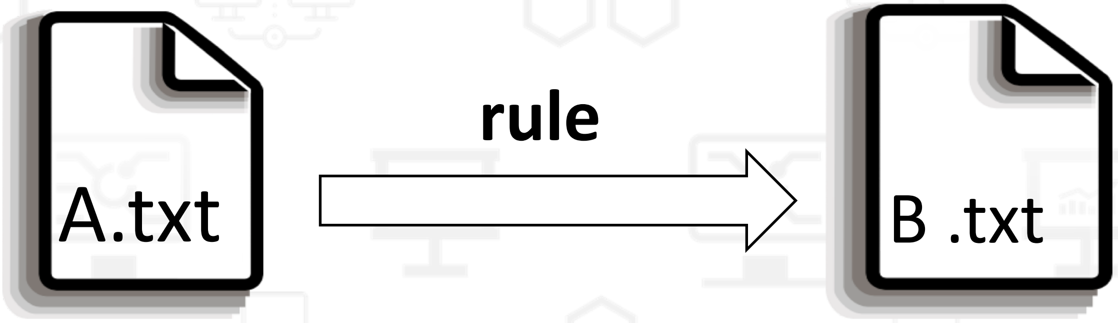


Management with scripts
(Bash, Perl, Python, ..)

Snakemake

基于Rule的规则，进行工作流程的构建

Demo: 拷贝文件



rule name

↑

```
rule copy:
      input:
            "A.txt"
      output:
            "B.txt"
      shell:
            "cp {input} {output}"
```

↓

通过shell，在输入文件和输出文件间建立联系

Snakemake

基于通配符规则，建立文件关联

Demo： sam转换成bam

利用通配的模式实现输入和输入文件关联

```
rule map:
    input:
        "{sample}.sam"
    output:
        "{sample}.bam"
    shell:
        "somecommand {input} {output}"
```

✗

Test.Cancer.markdup.bam
不能存在解析歧义
{sample}.{type}.bam

```
Sample = "Test"
Type = "Cancer.markdup"
          ???
Sample = "Test.Cancer"
Type = "markdup
```

Snakemake

多文件的input和output

Demo：　两个Bam的merge

输入和输出可以支持多个样本

```
rule map:
     input:
          "{sample}.L1.bam"
          "{sample}.L2.bam"
     output:
          "{sample}.merge.bam"
     shell:
          "somecommand {input[0]} {input[1]} {output}"
```

通过索引实现数据的引用。

Snakemake

多文件的input和output

Demo：　两个Bam的merge

多个不同的输入可以进行命名

```
rule map:
     input:
          a = “{sample}.L1.bam”
          b = “{sample}.L2.bam”
     output:
          “{sample}.merge.bam”
     shell:
          “somecommand {input.a} {input.b} {output}”
```

通过名称实现数据的引用

Snakemake

Input和output文件

Demo： 两个Bam的merge

```
rule samtools_index:
    input:
        "sorted_reads/{sample}.bam"
    output:
        "sorted_reads/{sample}.bam.bai"
    shell:
        "samtools index {input}"
```

Input、Output可以不出现在shell中

Snakemake

执行Python命令 替代 shell命令

Demo:　文本处理

```
rule sort:
    input:
        a = "path/to/{dataset}.txt"
    output:
        b = "{dataset}.sorted.txt"
    run:
        with open(output.b, "w") as out:
          for l in sorted(open(input.a)):
            print(l, file=out)
```

↓

Rule中执行Python语法

Snakemake

Rule中，引用其他(Python 或R)脚本

Demo： 文本处理

```
rule sort:
    input:
        a="calls/all.vcf"
    output:
        b="plots/quals.svg"
    script:
        "scripts/plot-quals.py"
```

引用Python 或 R 的脚本

scripts/plot-quals.py

```
import matplotlib
matplotlib.use("Agg")
import matplotlib.pyplot as plt
from pysam import VariantFile
quals = [record.qual for record in VariantFile(snakemake.input[0])]
plt.hist(quals)

plt.savefig(snakemake.output[0])
```

引用脚本与Snakemake的交互

Snakemake

多Rule之间的交互

Demo：收集多个结果

Python编码可以正常使用

```
DATASETS = ["D1", "D2", "D3"]

rule all:
    input:
        expand("{dataset}.sorted.txt",
dataset=DATASETS)


rule sort:
    input:
        "path/to/{dataset}.txt"
    output:
        "{dataset}.sorted.txt"
    shell:
        "sort {input} > {output}"
```

Job1：如果不指定Rule 则默认第一个出现的Rule

Job i: 应用rule sort来生成job1所需要的输入文件

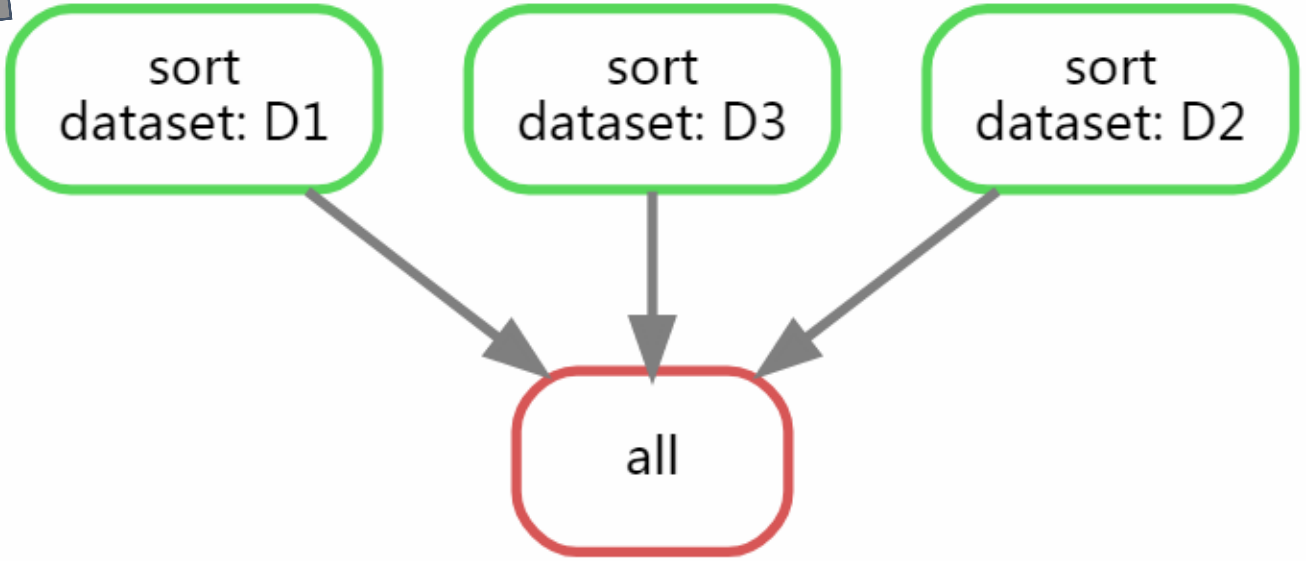
Snakemake

多Rule之间的交互

Demo： 收集多个结果

```
DATASETS = ["D1", "D2", "D3"]

rule all:
    input:
        expand("{dataset}.sorted.txt", dataset=DATASETS)


rule sort:
    input:
        "path/to/{dataset}.txt"
    output:
        "{dataset}.sorted.txt"
    shell:
        "sort {input} > {output}"
```

构建DAG图



Snakemake

Rule执行的标准

一个任务有且只有满足以下条件时才会被执行：

- ✓ 输出文件是目标文件，同时没有被生成过
- ✓ 输出文件是其他Rule的依赖文件，且没有被生成过
- ✓ 输入文件的创建时间比输出文件要晚（文件有更新）
- ✓ 输入文件会被其他任务更新
- ✓ 通过参数设置了强制执行
- ✓ 通过DAG确定的依赖Rule

Snakemake

简单命令行参数

```
# execute the workflow with target D1.sorted.txt
snakemake D1.sorted.txt
```

指定目标文件

```
# execute the workflow without target: first rule defines target
snakemake
```

默认执行第一个Rule

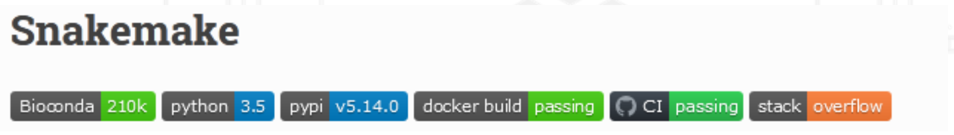
```
# dry-run
snakemake -n

# dry-run, print shell commands
snakemake -n -p

# dry-run, print execution reason for each job
snakemake -n -r

# visualize the DAG of jobs using the Graphviz dot command
snakemake --dag | dot -Tsvg > dag.svg
```

Snakemake Tutorial

1 基础规则介绍

2 **进阶使用**

3 扩展功能



Snakemake

Bioconda 210k | python 3.5 | pypi v5.14.0 | docker build passing | CI passing | stack overflow

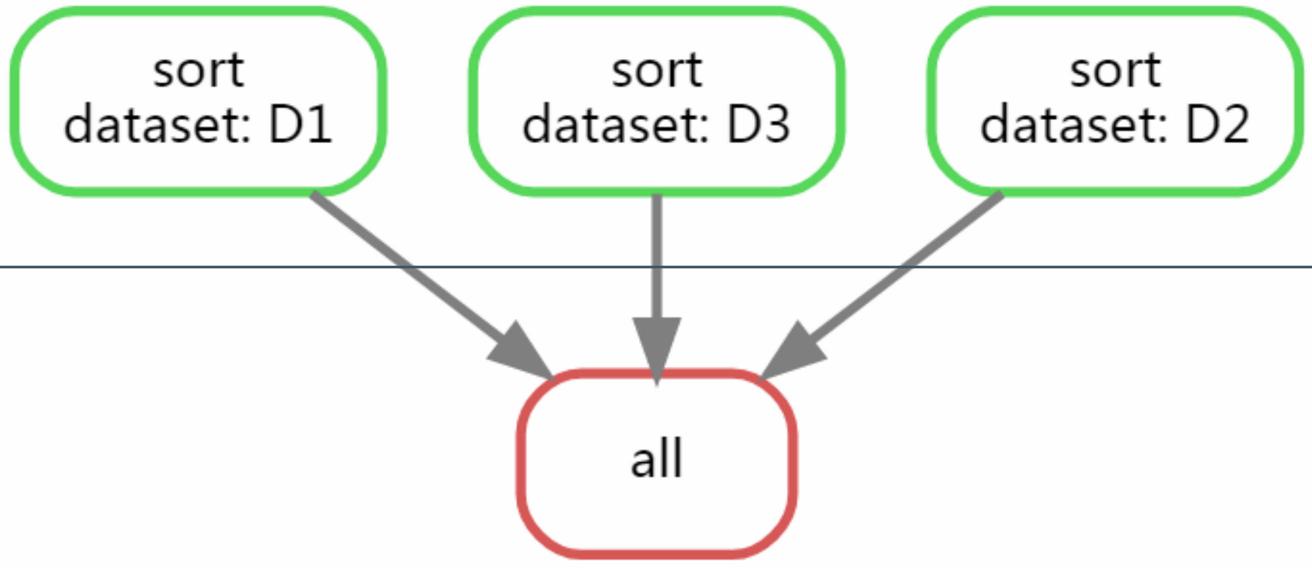
https://snakemake.readthedocs.io/

Snakemake

任务间的运行关系

Demo： 收集多个结果

独立任务
并行分析



```
# execute the workflow with 8 cores
snakemake --cores 8
```

Snakemake

在rule中资源进行资源设置

Demo： 收集多个结果

```
DATASETS = ["D1", "D2", "D3"]

rule all:
    input:
        expand("{dataset}.sorted.txt", dataset=DATASETS)


rule sort:
    input:
        "path/to/{dataset}.txt"
    output:
        "{dataset}.sorted.txt‟
    threads: 4
    resources: mem_mb=100
    shell:
        "sort --parallel {threads} {input} > {output}"
```

指定任务所需的资源 ←

↓ **Shell中指定资源**

Snakemake

命令行运行时的资源配置

Demo： 任务投递设置资源

```
DATASETS = ["D1", "D2", "D3"]

rule all:
    input:
        expand("{dataset}.sorted.txt", dataset=DATASETS)


rule sort:
    input:
        "path/to/{dataset}.txt"
    output:
        "{dataset}.sorted.txt"
    threads: 4
    resources: mem_mb=100
    shell:
        "sort --parallel {threads} {input} > {output}"
```

并行运行两个任务 **2 = 8/4**

```
# execute the workflow with 8 cores
snakemake --cores 8
```

针对特定任务调整资源配置

```
# prioritize the creation of a certain file
snakemake --prioritize D1.sorted.txt --cores 8
```

并行运行一个任务 **1 = 100/100**

```
# execute the workflow with 8 cores and 100MB memory
snakemake --cores 8 --resources mem_mb=100
```

Snakemake

配置文件的使用

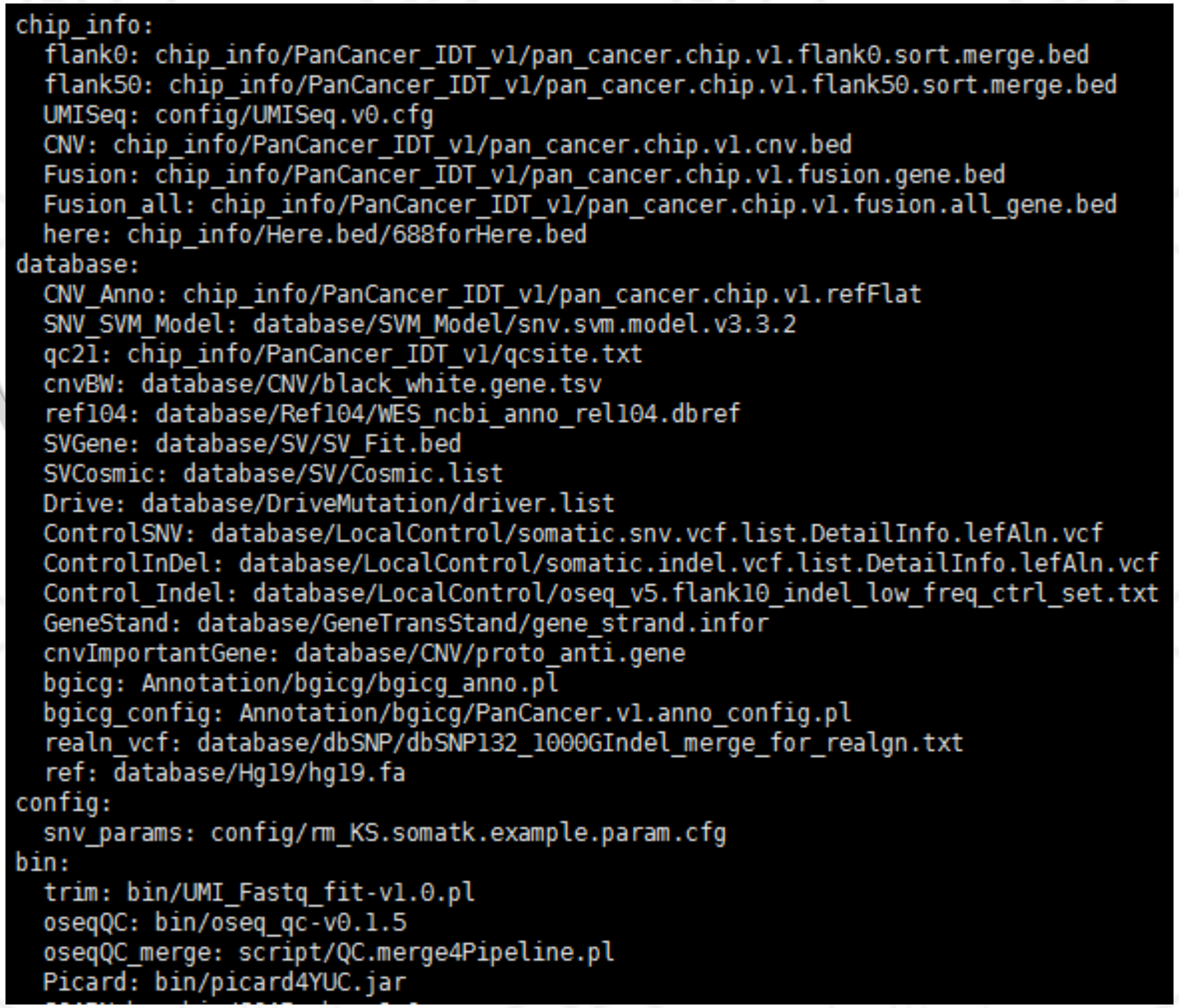
yaml格式:

缩进表示层级结构
冒号表示键值关系

key:{key1: value1, key2: value2, ...}。

```
key:
        key1:value1
        key2:value2
```

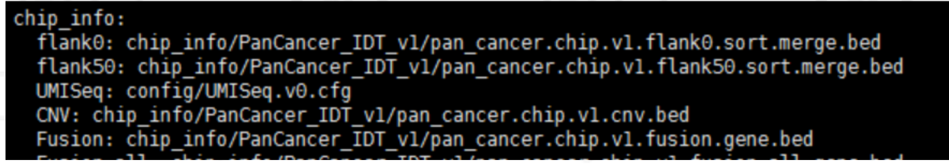
PanCancer.Pipeline.git.yaml



```
chip_info:
  flank0: chip_info/PanCancer_IDT_v1/pan_cancer.chip.v1.flank0.sort.merge.bed
  flank50: chip_info/PanCancer_IDT_v1/pan_cancer.chip.v1.flank50.sort.merge.bed
  UMISeq: config/UMISeq.v0.cfg
  CNV: chip_info/PanCancer_IDT_v1/pan_cancer.chip.v1.cnv.bed
  Fusion: chip_info/PanCancer_IDT_v1/pan_cancer.chip.v1.fusion.gene.bed
  Fusion_all: chip_info/PanCancer_IDT_v1/pan_cancer.chip.v1.fusion.all_gene.bed
  here: chip_info/Here.bed/688forHere.bed
database:
  CNV_Anno: chip_info/PanCancer_IDT_v1/pan_cancer.chip.v1.refFlat
  SNV_SVM_Model: database/SVM_Model/snv.svm.model.v3.3.2
  qc21: chip_info/PanCancer_IDT_v1/qcsite.txt
  cnvBW: database/CNV/black_white.gene.tsv
  ref104: database/Ref104/WES_ncbi_anno_rel104.dbref
  SVGene: database/SV/SV_Fit.bed
  SVCosmic: database/SV/Cosmic.list
  Drive: database/DriveMutation/driver.list
  ControlSNV: database/LocalControl/somatic.snv.vcf.list.DetailInfo.lefAln.vcf
  ControlInDel: database/LocalControl/somatic.indel.vcf.list.DetailInfo.lefAln.vcf
  Control_Indel: database/LocalControl/oseq_v5.flank10_indel_low_freq_ctrl_set.txt
  GeneStand: database/GeneTransStand/gene_strand.infor
  cnvImportantGene: database/CNV/proto_anti.gene
  bgicg: Annotation/bgicg/bgicg_anno.pl
  bgicg_config: Annotation/bgicg/PanCancer.v1.anno_config.pl
  realn_vcf: database/dbSNP/dbSNP132_1000GIndel_merge_for_realgn.txt
  ref: database/Hg19/hg19.fa
config:
  snv_params: config/rm_KS.somatk.example.param.cfg
bin:
  trim: bin/UMI_Fastq_fit-v1.0.pl
  oseqQC: bin/oseq_qc-v0.1.5
  oseqQC_merge: script/QC.merge4Pipeline.pl
  Picard: bin/picard4YUC.jar
```

Snakemake

配置文件的使用



```
chip_info:
  flank0: chip_info/PanCancer_IDT_v1/pan_cancer.chip.v1.flank0.sort.merge.bed
  flank50: chip_info/PanCancer_IDT_v1/pan_cancer.chip.v1.flank50.sort.merge.bed
  UMISeq: config/UMISeq.v0.cfg
  CNV: chip_info/PanCancer_IDT_v1/pan_cancer.chip.v1.cnv.bed
  Fusion: chip_info/PanCancer_IDT_v1/pan_cancer.chip.v1.fusion.gene.bed
```

```
configfile: "config.yaml"

rule all:
        expand("{dataset}.sorted.txt", dataset=config["chip_info"]["flank0"])


rule sort:
    input:
        "path/to/{dataset}.txt"
    output:
        "{dataset}.sorted.txt"
    threads: 4
    resources: mem_mb=100
    shell:
        "sort --parallel {threads} {input} > {output}"
```

→ 导入配置文件

↓ 引用配置文件

Snakemake

输入文件使用函数

Demo：bam文件的合并，并生成 Test.merge.bam

```
library_bams["Test"]=["test_a.bam","test_b.bam"]
```

→ 分析开始前基于数据更新变量

```
rule Library_Merge:
    input:
        lambda wildcards: library_bams[wildcards.library],
    output:
        "{library}.merge.bam",
    resources:
        mem_mb=10000,
    threads: 1
        shell:
                """
samtools merge {output} {input}
"""
```

↓ 基于分析需求获得对应数据

部分文件的匹配关系是一对多的，或者在流程构建阶段是不可知的，则可以使用通配符进行动态的匹配

Snakemake

更多的选择性关键字

Demo：Fq的比对

```
rule aln:
    input:
        "fq1"
    output:
        temp("{sample}.aln.bam")
       "{sample}.merge.bam"
    log:
        "log"
    conda:
        "envs/mapping.yaml"
    params:
        rg=r"@RG\tID:{sample}\tSM:{sample}"
    shell:
        command {input}  {output} > {log}
```

temp → 声明临时文件（临时文件会在使用后删除）

log → 声明log文件（和output的区别：分析失败后不会被删除）

conda → 声明每个rule所需要的环境（conda配置文件）

params → 提供流程的一些参数配置

Snakemake

更多的选择性关键字

Demo：Fq的比对及Merge

```
rule aln:
    input:
        "fq1"
    output:
        "{sample}.aln.bam"
       "{sample}.merge.bam"
    log:
        "log"
    conda:
        "envs/mapping.yaml"
    params:
        rg=r"@RG\tID:{sample}\tSM:{sample}"
    shell:
        command {input}  {output} > {log}
```

➡ 声明log文件（和output的区别：分析失败后不会被删除）

➡ 声明每个rule所需要的环境（conda配置文件）

➡ 提供流程的一些参数配置

Snakemake

临时文件与保护文件

Demo： 序列比对及排序

```
rule bwa_map:
    input:
        "data/genome.fa",
        lambda wildcards: config["samples"][wildcards.sample]
    output:
        temp("mapped_reads/{sample}.bam")
```

→ 声明临时文件（临时文件会在使用后删除）

```
    params:
        rg=r"@RG\tID:{sample}\tSM:{sample}"
shell:
        "(bwa mem -R '{params.rg}' {input} |samtools view -Sb - > {output}) 2> "

rule samtools_sort:
    input:
        "mapped_reads/{sample}.bam"
    output:
        protected("sorted_reads/{sample}.bam")
```

→ 声明保护文件（分析后权限改为 **444** ）

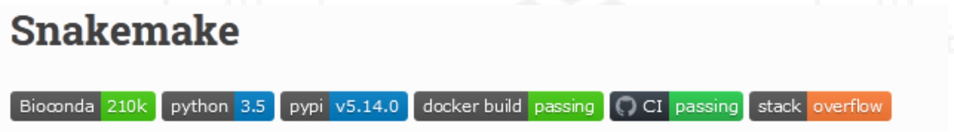
```
    shell:
        "samtools sort -T sorted_reads/{wildcards.sample} "
        "-O bam {input} > {output}"
```

Snakemake Tutorial

1 基础规则介绍

2 进阶使用

3 **扩展功能**



Snakemake

Bioconda 210k | python 3.5 | pypi v5.14.0 | docker build passing | CI passing | stack overflow

https://snakemake.readthedocs.io/

Snakemake

Benchmark监控Rule资源

Demo：Fq的比对

```
rule aln:
    input:
        "fq1"
    output:
        temp("{sample}.aln.bam")
    log:
        "log"
    benchmark:
        repeat("benchmarks/{sample}.bwa.benchmark.txt", 5)
    params:
        rg=r"@RG\tID:{sample}\tSM:{sample}"
    shell:
        command {input}  {output} > {log}
```

→ 指定记录资源消耗的文件，重复5次

benchmark.txt

s	h:m:s	max_rss	max_vms	max_uss	max_pss	io_in	io_out	mean_load
1.927	0:00:01	1.18	103.64	0.19	0.2	0	0.2	0
1.9574	0:00:01	1.19	103.64	0.19	0.21	0	0.19	0
1.7014	0:00:01	1.19	103.64	0.19	0.2	0	0.22	0.62
2.1334	0:00:02	1.19	103.64	0.19	0.2	0	0.23	0.46
2.416	0:00:02	1.18	103.64	0.19	0.2	0	0.19	0.41

Snakemake

rule的模块化

Demo： Fq的比对

```
rule aln:
    input:
        "fq1"
    output:
        temp("{sample}.aln.bam")
    log:
        "log"
    benchmark:
        repeat("benchmarks/{sample}.bwa.benchmark.txt", 5)
    params:
        rg=r"@RG\tID:{sample}\tSM:{sample}"
    shell:
        command {input}  {output} > {log}
```

aln.smk

```
include: "aln.smk"
```

通过include关键字，可以在snakemake中调用不同的rule实现复用。

```
include: "Single.snv.smk"
include: "Pair.snv.smk"
```

Snakemake

软件依赖关系的自动部署

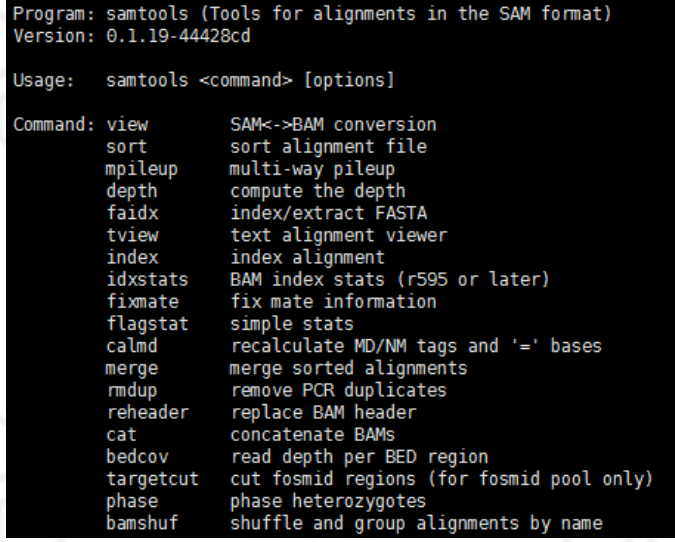
Demo： 构建索引

```
rule samtools_index:
  input:
      "sorted_reads/{sample}.bam"
  output:
      "sorted_reads/{sample}.bam.bai"
  conda:
      "envs/samtools.yaml"
  shell:
      "samtools index {input}"
```

```
channels:
  - bioconda
dependencies:
  - samtools =1.9
```

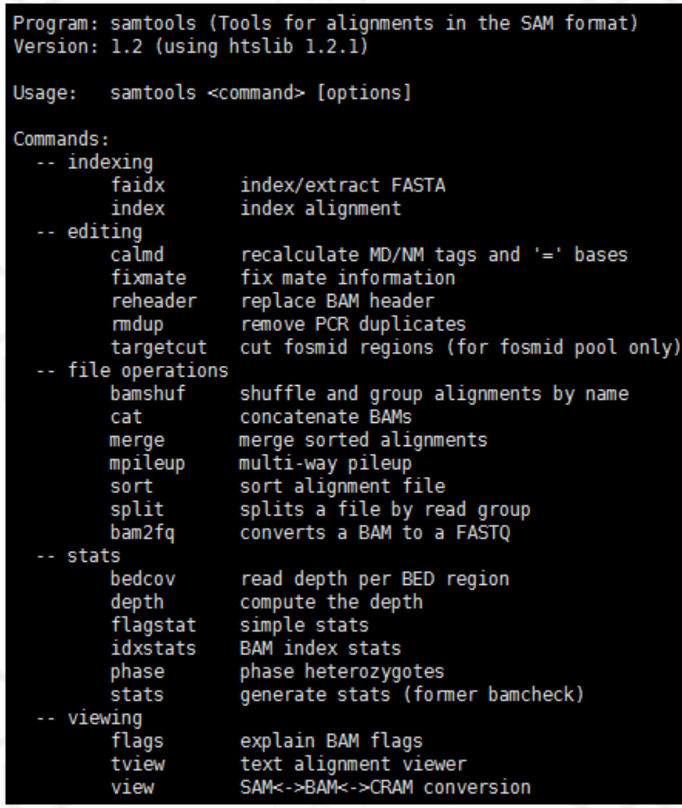
envs/samtools.yaml

Samtools-0.1.19



```
Program: samtools (Tools for alignments in the SAM format)
Version: 0.1.19-44428cd

Usage:   samtools <command> [options]

Command: view        SAM<->BAM conversion
         sort        sort alignment file
         mpileup     multi-way pileup
         depth       compute the depth
         faidx       index/extract FASTA
         tview       text alignment viewer
         index       index alignment
         idxstats    BAM index stats (r595 or later)
         fixmate     fix mate information
         flagstat    simple stats
         calmd       recalculate MD/NM tags and '=' bases
         merge       merge sorted alignments
         rmdup       remove PCR duplicates
         reheader    replace BAM header
         cat         concatenate BAMs
         bedcov      read depth per BED region
         targetcut   cut fosmid regions (for fosmid pool only)
         phase       phase heterozygotes
         bamshuf     shuffle and group alignments by name
```

Samtools-1.2



```
Program: samtools (Tools for alignments in the SAM format)
Version: 1.2 (using htslib 1.2.1)

Usage:   samtools <command> [options]

Commands:
  -- indexing
         faidx       index/extract FASTA
         index       index alignment
  -- editing
         calmd       recalculate MD/NM tags and '=' bases
         fixmate     fix mate information
         reheader    replace BAM header
         rmdup       remove PCR duplicates
         targetcut   cut fosmid regions (for fosmid pool only)
  -- file operations
         bamshuf     shuffle and group alignments by name
         cat         concatenate BAMs
         merge       merge sorted alignments
         mpileup     multi-way pileup
         sort        sort alignment file
         split       splits a file by read group
         bam2fq      converts a BAM to a FASTQ
  -- stats
         bedcov      read depth per BED region
         depth       compute the depth
         flagstat    simple stats
         idxstats    BAM index stats
         phase       phase heterozygotes
         stats       generate stats (former bamcheck)
  -- viewing
         flags       explain BAM flags
         tview       text alignment viewer
         view        SAM<->BAM<->CRAM conversion
```

通过每个rule设置环境变量，从而消除版本、包依赖带来的结果影响

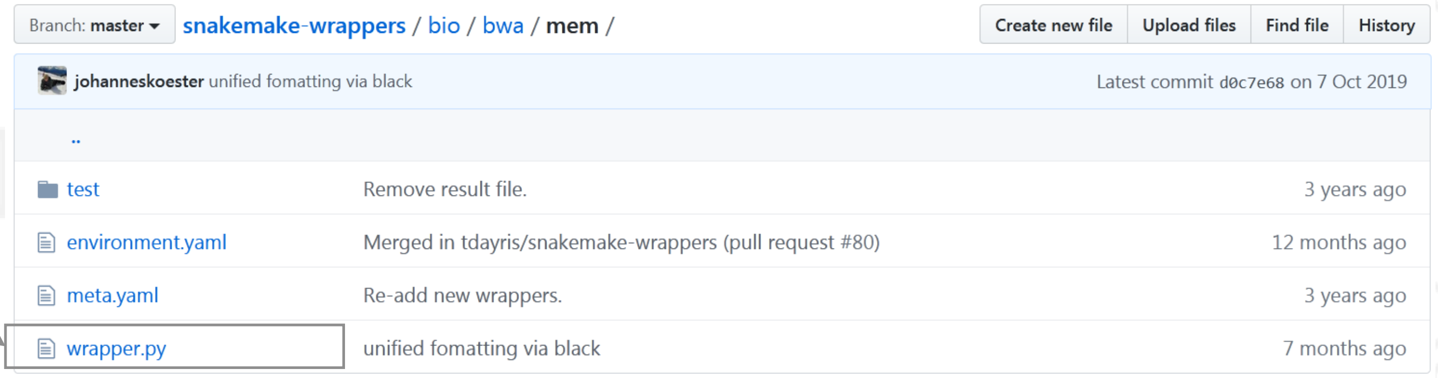
Snakemake

利用先用的封装包

Demo: bam文件的合并，并生成 Test.merge.bam

```
rule bwa_mem:
  input:
      ref="data/genome.fa",
      sample=lambda wildcards: config["samples"][wildcards.sample]
  output:
      temp("mapped_reads/{sample}.bam")
  log:
      "logs/bwa_mem/{sample}.log"
  params:
      "-R '@RG\tID:{sample}\tSM:{sample}'"
  threads: 8
  wrapper:
      "0.15.3/bio/bwa/mem"
```

使用已经发布的工具集


Branch: master ▾ snakemake-wrappers / bio / bwa / mem /

Create new file | Upload files | Find file | History

johanneskoester unified fomatting via black — Latest commit d0c7e68 on 7 Oct 2019

..

test	Remove result file.	3 years ago
environment.yaml	Merged in tdayris/snakemake-wrappers (pull request #80)	12 months ago
meta.yaml	Re-add new wrappers.	3 years ago
wrapper.py	unified fomatting via black	7 months ago

https://github.com/snakemake/snakemake-wrappers/tree/master/bio

Snakemake

多个rule的规则优先级排序

Demo：bam文件的合并，并生成 Test.merge.bam

```
rule all:
  input:
    "out"
rule A :
  input:
    "a"
  output:
    "out"
  shell:
    "touch out"
rule B :
  input:
    "a","c"
  output:
    "out"
  shell:
    "touch out`
```

输出结果冲突

```
ruleorder:   B > A
```

$\longrightarrow$ 决定**Rule**执行的优先级

Snakemake

模糊规则处理

Demo：bam文件的合并，并生成 Test.merge.bam

```
# a target rule to define the desired final output
rule all:
    input:
        "aggregated/a.txt",
        "aggregated/b.txt"
# intermediate rule
rule intermediate:
    input:
        "somestep/{sample}.txt"
    output:
        "post/{sample}.txt"
    shell:
        "touch {output}"
# alternative intermediate rule
rule alt_intermediate:
    input:
        "somestep/{sample}.txt"
    output:
        "alt/{sample}.txt"
    shell:
        "touch {output}"


# input function for the rule aggregate
def aggregate_input(wildcards):
    # decision based on content of output file
    # Important: use the method open() of the returned file!
    # This way, Snakemake is able to automatically download the file if it is generate
    # a cloud environment without a shared filesystem.
    with checkpoints.somestep.get(sample=wildcards.sample).output[0].open() as f:
        if f.read().strip() == "a":
            return "post/{sample}.txt"
        else:
            return "alt/{sample}.txt"

rule aggregate:
    input:
        aggregate_input
    output:
        "aggregated/{sample}.txt"
    shell:
        "touch {output}"
```

Snakemake

集群任务的投递

```
# execute the workflow on cluster with qsub submission command
# (and up to 100 parallel jobs)
snakemake --cluster qsub --jobs 100
# tell the cluster system about the used threads
snakemake --cluster "qsub -pe threaded {threads}" --jobs 100

snakemake
--restart-times#失败后重复投递的次数
--jobs          #设置并行的最大任务数目。
--config        #向snakemake传递参数（字典形式）
--configfile    #指定配置文件路径（可以支持多个）
--cores         #设置任务最多使用的核数
--resources     #设置任务最多使用的内存
--touch         #更新文件的时间戳（不会重新跑）
--keep-going    #一个任务失败后，其他独立任务继续运行
--force         #强制执行某一条
--forceall      #强制执行某条Rule及它的依赖。
--forcerun      #强制执行某条Rule，并更新后续依赖它的Rule。
--dry-run       #生成运行的shell逻辑结构，但是不投递任务
--unlock        #解锁目录，任务投递后但是没有正常结束的时候目录会被锁，重新投递需要先解锁
--list          #展示smk脚本中所能获得的所有Rule
--dag           #生成整个流程的有向无环图，但不进行分析
```

Snakemake

Solution 1: Git repository with

```
├── config.yaml
├── requirements.txt
├── scripts
│   ├── script1.py
│   └── script2.R
└── Snakefile
```

```
# clone workflow into working directory
git clone https://bitbucket.org/user/myworkflow.git path/to/workdir
cd path/to/workdir

# edit config and workflow as needed
vim config.yaml

# install dependencies into isolated environment
conda create -n myworkflow --file requirements.txt

# activate environment
source activate myworkflow

# execute workflow
snakemake -n
```

Thank you