Snakemake Presentation

Dimitri Desvillechabrol and Thomas Cokelaer

Institut Pasteur

March 22 2016

Workflow management

- Many methods to implement a workflow.
- They must handle:
  - Parallelization
  - Suspend/Resume

Workflow management

- Many methods to implement a workflow.
- They must handle:
  - Parallelization
  - Suspend/Resume

Snakemake

- Workflow management system based on python.
- Simplify the workflow's conceptions.
- Decompose workflow into rules.

Basic rule

Snakefile

```
rule bwa_mapping:
    input:
        ref = "genome.fa",
        fastq = "sample_A.fastq.gz"
    output:
        "mapped_sample/sample_A.bam"
    shell:
        "bwa mem {input.ref} {input.fastq} | samtools view -Sb - > {output}"
```

Basic rule

Snakefile

```
rule bwa_mapping:
    input:
        ref = "genome.fa",
        fastq = "sample_A.fastq.gz"
    output:
        "mapped_sample/sample_A.bam"
    shell:
        "bwa mem {input.ref} {input.fastq} | samtools view -Sb - > {output}"
```

Shell

```
$ snakemake
Job counts:
        count   jobs
        1       bwa_mapping
        1
rule bwa_mapping:
        input: genome.fa, sample_A.fastq.gz
        output: mapped_sample/sample_A.bam
1 of 1 steps (100%) done
$ snakemake
Nothing to be done.
```

Generalizing the rule

Snakefile

```
SAMPLE = ["sample_A", "sample_B"]

rule all:
    input:
        expand("mapped_sample/{sample}.bam", sample=SAMPLE)

rule bwa_mapping:
    input:
        ref = "genome.fa",
        fastq = "{sample}.fastq"
    output:
        "mapped_sample/{sample}.bam"
    shell:
        "bwa mem {input.ref} {input.fastq} | samtools view -Sb - > {output}"
```

Adding a rule

Snakefile

```
SAMPLE = ["sample_A", "sample_B"]

rule all:
    input:
        expand("sorted/{sample}.bam", sample=SAMPLE)

rule bwa_mapping:
    input:
        ref = "genome.fa",
        fastq = "{sample}.fastq"
    output:
        "mapped_sample/{sample}.bam"
    shell:
        "bwa mem {input.ref} {input.fastq} | samtools view -Sb - > {output}"

rule sort_mapping:
    input:
        bam = "mapped_sample/{sample}.bam"
    output:
        "sorted/{sample}.bam"
    shell:
        "samtools sort -O bam {input.bam} > {output}"
```

Config file

- We can add a config file (JSON/YAML) in our Snakefile.

config.yaml

```
samples:[sample_A, sample_B]
```

Snakefile

```
configfile: "config.yaml" # Ou .json

rule all:
    input:
        expand("mapped_sample/{sample}.bam", sample=config["samples"])

rule bwa_mapping:
    input:
        ref = "genome.fa",
        fastq = "{sample}.fastq"
    output:
        "mapped_sample/{sample}.bam"
    shell:
        "bwa mem {input.ref} {input.fastq} | samtools view -Sb - > {output}"
```

YAML file

config_vc.yaml

```
reference: reference.fa

sample:
    - ERR036019.bam

markduplicate: true

freebayes:
    --ploidy: 1

vcf_filter:
    QUAL: 10000
    FREQ: 0.9
    INFO:
        DP: ">10"
```

Optional rule

Snakefile

```
MARKJOB = config["markduplicate"]
MARKTAG = ""
if MARKJOB:
    MARKTAG = "_undup"

rule markDuplicate:
    input:
        bam = "{sample}.bam"
    output:
        a = "{sample}%s.bam" % MARKTAG,
        b = temp("{sample}.metrics")
    run:
        if MARKJOB:
            shell("./MarkDuplicates I={input.bam} O={output.a} M={output.b}")
        else:
            shell("touch {output.b}")
```

- NB: We can't use the ".format()" method.

Optional parameter

Snakefile

```
rule samtools_index:
    input:
        bam = "{sample}%s.bam" % MARKTAG
    output:
        "{sample}%s.bam.bai" % MARKTAG
    shell:
        "samtools index {input.bam}"

rule freebayes:
    input:
        bam = "{sample}%s.bam" % MARKTAG,
        bai = "{sample}%s.bam.bai" % MARKTAG,
        ref = config["reference"]
    output:
        vcf = "{sample}.vcf"
    params:
        " ".join(['%s %s' % (key, value) for (key, value) in \
            config["freebayes"].items()])
    shell:
        "freebayes {params} -f {input.ref} -b {input.bam} -v {output.vcf}"
```

Running python code

Snakefile

```
rule vcf_filter:
    input:
        vcf = "{sample}.vcf"
    output:
        vcf = "{sample}_filter.vcf"
    run:
        from sequana import vcf_filter
        vcf_record = vcf_filter.VCF(input["vcf"])
        vcf_record.filter_vcf(config["vcf_filter"], output["vcf"])
```

Running snakemake

- If all your files are in the current directory:

```
snakemake
```

- We can specify the path of Snakefile or a config file:

```
snakemake -s path/to/Snakefile --configfile path/to/configfile
```

- If you want to run on 4 cores:

```
snakemake -cores 4
```

Interesting option

- Dry-run is able to test if your Snakefile is valid:

```
snakemake -n
```

- Change config variable in command line:

```
snakemake --config markduplicates=false
```

Doesn't work with boolean parameter because it becomes a string.

- Running only one rule, here freebayes rule:

```
snakemake --force freebayes
```

Doesn't work if the rule uses wildcards declared in other rule.

```
RuleException in line 42 of /home/desvillechabrol/Documents/pasteur/
variant_calling/Snakefile:
Could not resolve wildcards in rule freebayes:
sample
```

Running snakemake on bic

- It is very simple to run a workflow on bic:

```
module load snakemake
/local/gensoft2/exe/snakemake/3.5.4/bin/snakemake -p --cluster \
"qsub -q pf4 -cwd -V -b y" --jobs 5
```

- Module load must be added inside rules.

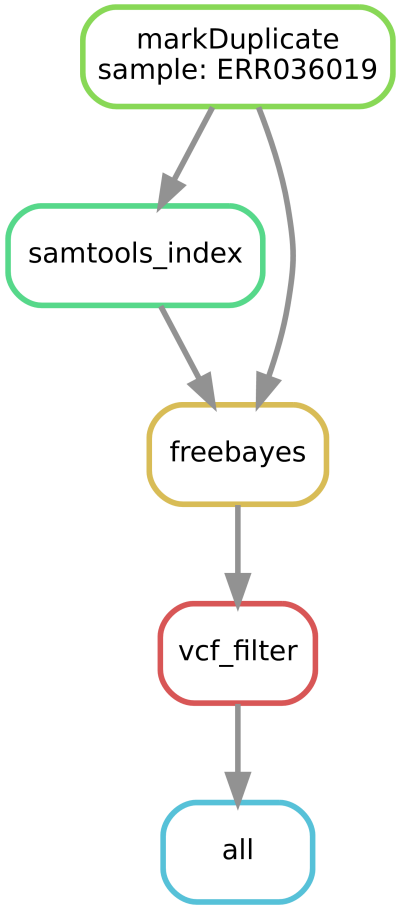
Snakefile

```
rule bwa_mapping:
    input:
        ref = "genome.fa",
        fastq = "sample_A.fastq.gz"
    output:
        "mapped_sample/sample_A.bam"
    shell:
        """
        . /etc/profile.d/pasteur_modules.sh
        module load bwa/0.7.5a
        bwa mem {input.ref} {input.fastq} | samtools view -Sb - > {output}
        """
```

Directed Acyclic Graph (DAG)

- We can generate the DAG of your workflow:

```
snakemake --dag | dot -Tsvg > dag.svg
```



References

- Snakemake website:
  - https://bitbucket.org/snakemake/snakemake/wiki/Home
- Useful information:
  - http://slides.com/johanneskoester/deck-1
  - http://snakemake.bitbucket.org/snakemake-tutorial.html
  - http://slowkow.com/notes/snakemake-tutorial/
  - http://watson.nci.nih.gov/~sdavis/blog/flexible_bioinformatics_pipelines_with_snakemake/
- Advanced example:
  - https://github.com/sequana/sequana/tree/master/pipelines/variants
- Enable syntax in vim for Snakemake
  - http://tinyurl.com/zm4b6c4

Additional keywords

threads Set the number of threads needed

log Set a log file for a rule

Snakefile

```
rule bwa_mapping:
    input:
        ref = "genome.fa",
        fastq = "sample_A.fastq.gz"
    output:
        "mapped_sample/sample_A.bam"
    threads: 8
    log:
        "logs/bwa_mapping/sample_A.log"
    shell:
        """
        (bwa mem -t {threads} {input.ref} {input.fastq} | \
        samtools view -Sb - > {output}) 2> {log}
        """
```

Cluster special features

- Snakemake generates logs file with stdout and stderr of your jobs

- Your currents jobs are displaying with qstat

```
# ddesvill@bic ~/Test_fastq_dir > qstat
job-ID  prior   name       user         state submit/start at
queue                          slots ja-task-ID
-----------------------------------------------------------------------------
-----------------------------------
3781631 0.63541 snakejob.b ddesvill     r     03/23/2016 12:17:04
pf4@bic-n119.cluster.pasteur.f     1
3781632 0.63541 snakejob.b ddesvill     r     03/23/2016 12:17:04
pf4@bic-n119.cluster.pasteur.f     1
3781633 0.63541 snakejob.b ddesvill     r     03/23/2016 12:17:04
pf4@bic-n106.cluster.pasteur.f     1
3781634 0.63541 snakejob.b ddesvill     r     03/23/2016 12:17:04
pf4@bic-n422.cluster.pasteur.f     1
3781635 0.63541 snakejob.b ddesvill     r     03/23/2016 12:17:04
pf4@bic-n422.cluster.pasteur.f     1
```

What happens when the snakemake is interrupted

- If you stop your snakemake (i.e. ctrl+c):

```
Terminating processes on user request.
Will exit after finishing currently running jobs.
Removing output files of failed job samtools since they might be corrupted:
reference.fa.fai
```

- On the cluster, the current job is not kill
- If you close your shell (Crash simulation):

```
IncompleteFilesException:
The files below seem to be incomplete. If you are sure that certain files are
not incomplete, mark them as complete with

    snakemake --cleanup-metadata <filenames>

To re-generate the files rerun your command with the --rerun-incomplete flag.
Incomplete files:
ERR036019_unsort.bam
```

- We can rerun the snakemake with `--rerun-incomplete`.