Workflow management with Snakemake

S. Rao

Outline

- Constructing a workflow to preprocess sequencing data
  - Fastq > align with bwa > sort > mark dups > plot duplication metrics
  - Covers: conda and environment modules, passing input/output/params to R/Python scripts
- Input functions
  - Covers: custom functions for input/params arguments

```
Python/snakemake code
```

```
Terminal commands/arguments
```

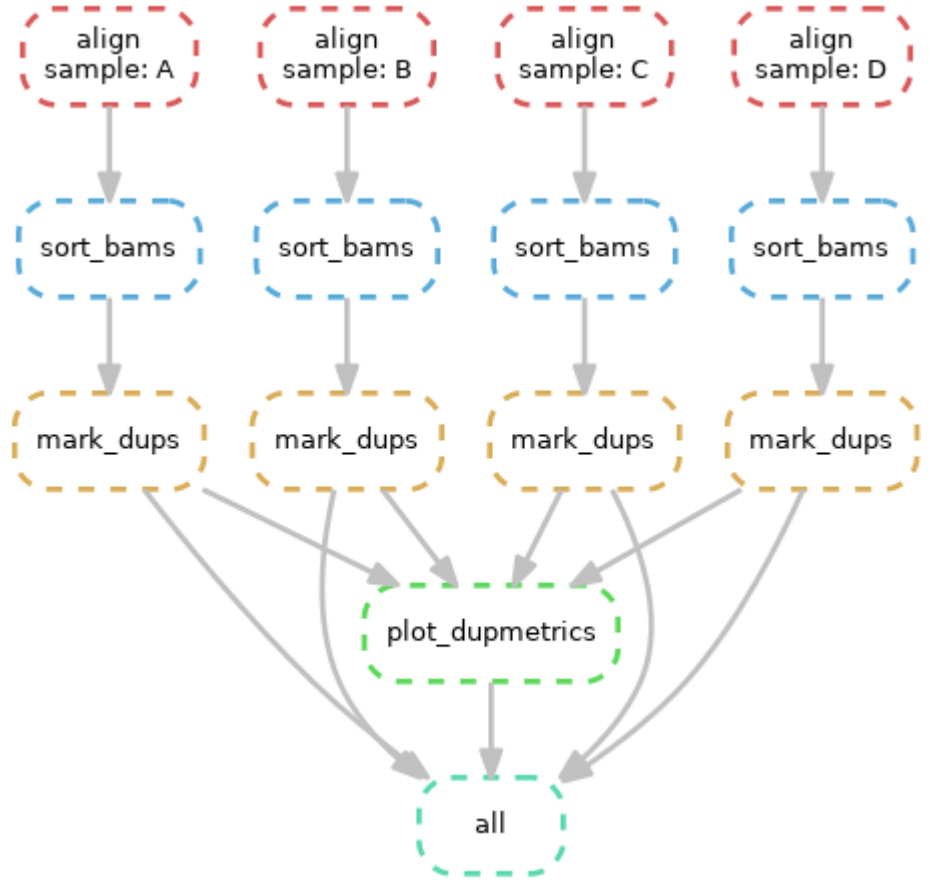
A simple snakemake rule

```
snakemake --snakefile Snakefile_cluster.smk --profile profile/
```

```
rule get_hostname:
    input: "Snakefile_cluster.smk"  # dummy input
    output: "hostname.txt"
    shell: "hostname > {output}"
```

A typical workflow

```
snakemake --snakefile Snakefile --use-conda --use-envmodules --profile profile/
```



Conda environments (per-rule)

envs/bwa.yaml

```
channels:
    - conda-forge
    - bioconda
dependencies:
    - bwa=0.7.17
```

Pros:

- Fully reproducible
- Software versions
- No clash of dependencies (if conda environments are minimal)

Cons:

- First run takes time to install conda env
- BMRC-specific – conda envs built using submission node but may raise ‘illegal instruction’ error when cpu architecture doesn’t match (e.g. C and D nodes)

Environment modules (per-rule)

```
envmodules: "R/default"
```

Pros:

- BMRC-specific – software matched to cpu architecture, so can be submitted to any node
- Quicker to start

Cons:

- Not reproducible on another system
- Software you want may not be installed

Best of both worlds?

```
snakemake --snakefile Snakefile --use-conda --use-envmodules --profile profile/
```

- Both Conda and envmodules can be specified
  - Envmodules used where specified
  - conda used when envmodules not specified
  - Where required, rule-specific settings can be changed to account for cpu architecture

Passing input/output/params to scripts - R

- R script invoked by snakemake contains a snakemake S4 object
- Accessed by

```
snakemake@input     # list
snakemake@output
snakemake@params
snakemake@threads
snakemake@config
```

- Example:

```
inputfile <- snakemake@input[[1]]
gatk_param <- snakemake@params[[“gatk”]]
```

Passing input/output/params to scripts - Python

- Python script invoked by snakemake contains a snakemake object
- Accessed by

```
snakemake.input    # list
snakemake.output
snakemake.params
snakemake.threads
snakemake.config
```

- Example:

```
inputfile = snakemake.input[0]
gatk_param = snakemake.params[“gatk”]
```

Input functions

- A function to generate a list of input files
  - Takes wildcards as argument
  - Function returns a list or dict
  - Can be lambda function

```
def get_ref(wildcards):
    if config["REF_VERSION"] == 37:
        return [config["REF37"]]
    elif config["REF_VERSION"] == 38:
        return [config["REF38"]]
    else:
        print("incorrect value for reference!")
```

Workflow visualisation

- Uses graphviz
  - Install in the snakemake environment

```
mamba install -c anaconda graphviz
```

- --dag option generates a text file that can be read by graphviz

```
snakemake --dag | dot -Tpng > test.png
```