Bioinformatics Workflows with Snakemake

Johannes Köster
Genome Informatics, Institute of Human Genetics, University of Duisburg-Essen
Paediatric Oncology, University Hospital Essen

CONTRIBUTION

In most bioinformatics analyses, many different files are created by the combination of various tools. A sophisticated workflow management eases the inclusion of new datasets and parameter changes and ensures reproducibility. Snakemake is a workflow engine that combines an easy to read text-based definition language with a versatile execution environment that is scalable from single core machines over multi-core servers to clusters.

IDEA

In Snakemake, a workflow is defined in terms of rules, that create output files from input files by applying a command line tool or executing Python or R:

define your workflow in terms of rules that create output from input files

collect your results with target rules

schedule intermediate files for cleanup

write-protect important results

invoke shell commands

use wildcards for generalization

script in Python or R

assign names to input and output files

```
DATASETS = ["552", "553", "554", "555"]

rule all:
    input: expand("expressions/{ds}.expressions.txt", ds=DATASETS),
           "stats/mirnareads.txt"

rule map_against_mirbase:
    input:    ref="mirbase/embedded-mirnas.fa",
              reads="reads/{ds}.fastq.gz"
    output:   temp("mapped/{ds}.bam")
    threads: 4
    shell: "sqt-bwa -t {threads} -r {input.ref} {output} {input.reads}"

rule sort_bam:
    input:   "{prefix}.bam"
    output: protected("{prefix}.sorted.bam")
    shell: "samtools sort {input} {wildcards.prefix}.sorted"

rule count_mirna_expressions:
    input:   "mapped/{ds}.sorted.bam"
    output: "expressions/{ds}.expressions.txt"
    shell:   "sqt-bamstats --quality 1 --maximum-start 3 {input} >> {output}"

rule count_reads_with_mirna_length:
    input:   expand("reads/{ds}.fastq.gz", ds=DATASETS)
    output: txt="stats/mirnareads.txt"
    run:
        with open(output.txt, "w") as out:
            for f in input:
                mirna_reads = sum(count
                    for length, count in readlength_histogram(f)
                    if MIRNA_MIN <= length <= MIRNA_MAX)
                print(wildcards.ds, mirna_reads, file=out)
```

FEATURES

Run the whole workflow (no arguments) or create specific targets (rule or filenames):

```
$ snakemake mapped/552.bam
```

Execute jobs in parallel up to a given limit of usable cores:

```
$ snakemake --cores 40
```

To run Snakemake on a cluster, only a shared filesystem and a qsub-like submission command is needed. Here, parameters of the rules can be forwarded to the clustering engine:

```
$ snakemake --cluster "qsub -pe threaded {threads}"
```

Summarize all involved files, listing pending updates based on modification dates, tool versions and implementation changes:

```
$ snakemake --summary | less
```

Visualize the workflow via graphviz dot:

```
$ snakemake --dag | dot | display
```

Force certain rules to re-run:

```
$ snakemake --forcerun count_mirna_expressions
```

Prioritize a target in the scheduler (for example an urgent dataset):

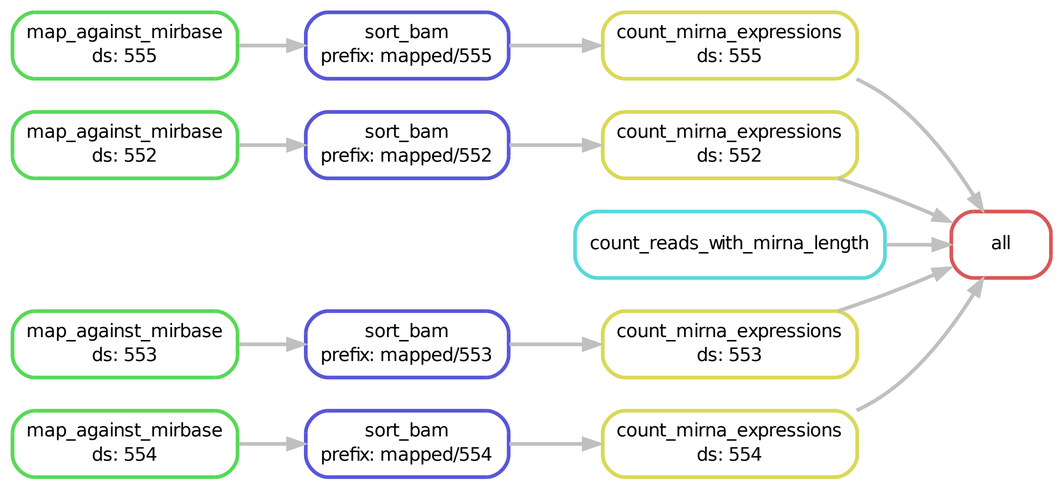
```
$ snakemake --prioritize expressions/552.expressions.txt
```

Finally, Snakemake provides facilities for

- dynamic updates depending on a rule's output,
- portable HTML5 reports that provide a semantically connected collection of generated figures and tables for collaborators,
- creation of rule libraries that can be included into workflows.

FROM RULES TO JOBS

Dependencies between the rules are determined automatically by matching input files against output files. This yields a directed acyclic graph of jobs (DAG) that represents the execution plan of Snakemake. Jobs are only executed if their output files are not present, at least one input file is newer that an output file, or if execution is forced via the command line.



Jobs that lie on the same path are dependent on each other. Independent jobs can be executed in parallel. Snakemake provides a scheduling mechanism that maximizes the usage of given cores with respect to threads, priorities and input file sizes by solving a knapsack problem.

Johannes Köster, Sven Rahmann. "Snakemake - A scalable bioinformatics workflow engine". Bioinformatics 2012.

visit Snakemake at
https://bitbucket.org/johanneskoester/snakemake