Introduction to snakemake

A workflow management system for bioinformatics

Verena Kutschera (verena.kutschera@scilifelab.se)

National Bioinformatics Infrastructure Sweden (NBIS)
Science for Life Laboratory Solna & Dept. for Biochemistry and Biophysics, Stockholm university

2019-09-03

Workflow management systems

SciLifeLab

Galaxy

COMMON WORKFLOW LANGUAGE

Snakemake

nextflow

luigi

bpipe

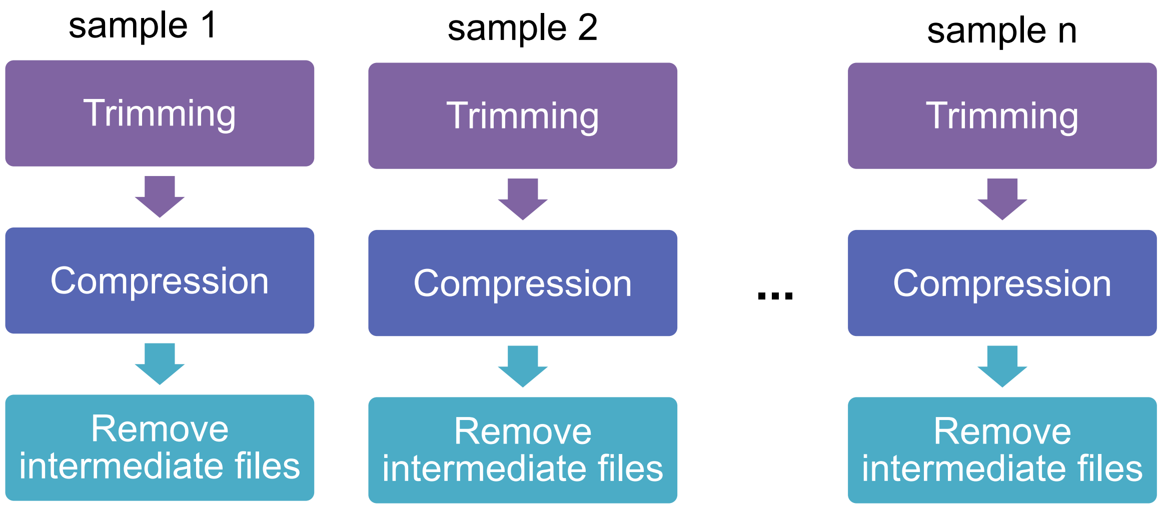
Ruffus

Leipzig (2017). A review of bioinformatic pipeline frameworks. Briefings in Bioinformatics, 18(3), 530–536.

Why workflow managers?

- Bash code

```
for sample in $(ls *.fastq | sed 's/.fastq//')
do
   # Trim fastq file
   trim_galore --illumina ${sample}.fastq > ${id}.trimmed.fastq

   # Compress fastq file
   gzip -c ${sample}.trimmed.fastq > ${sample}.trimmed.fastq.gz

   # Remove intermediate files
   rm ${sample}.trimmed.fastq
done
```



Why workflow managers?

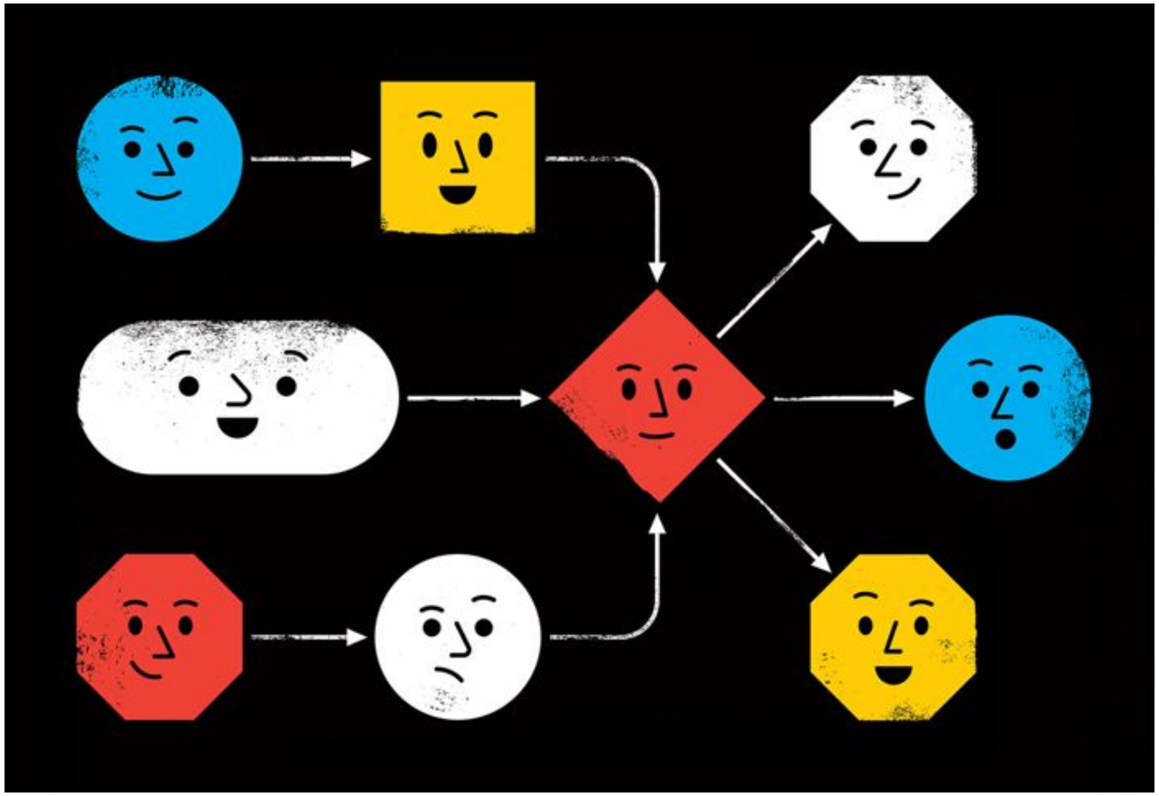
- What kinds of problems have you encountered when running your analyses using bash scripts similar to the example?

Why workflow managers?

- Interruption → continue where left off
- New samples → do not rerun everything
- Updated dependency → update downstream files, too

Why workflow managers?

“There is a learning curve to adopting workflow languages. But, [...] “the energy that you expend learning is more than made up for by the energy you save in having your code be reproducible.” (Brian Naughton)”



https://www.nature.com/articles/d41586-019-02619-z

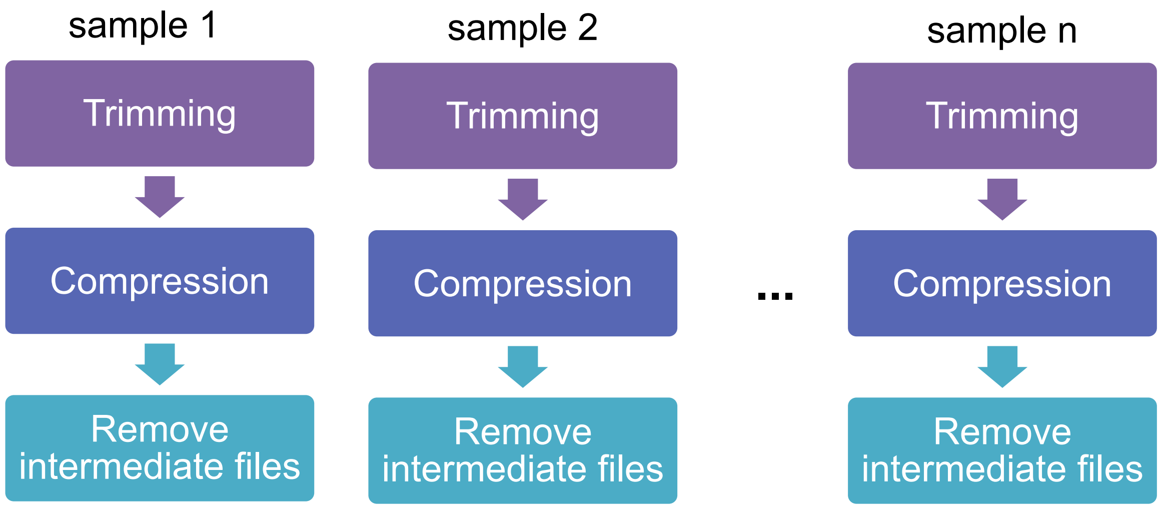
Snakemake

Python + GNU Make = Snakemake

Why workflow managers?

- Bash code

```
for sample in $(ls *.fastq | sed 's/.fastq//')
do
   # Trim fastq file
   trim_galore --illumina ${sample}.fastq > ${id}.trimmed.fastq

   # Compress fastq file
   gzip -c ${sample}.trimmed.fastq > ${sample}.trimmed.fastq.gz

   # Remove intermediate files
   rm ${sample}.trimmed.fastq
done
```



Snakemake

```
rule trim_fastq:
    input: "{prefix}.fastq"
    output: temp("{prefix}.trimmed.fastq")
    shell:
        "trim_galore --illumina {input} > {output}"

rule gzip:
    input: "{prefix}"
    output: "{prefix}.gz"
    shell:
        "gzip -c {input} > {output}"
```

```
$ snakemake {a,b}.trimmed.fastq.gz
Building DAG of jobs...
Using shell: /bin/bash
Provided cores: 1
Rules claiming more threads will be scaled down.
Job counts:
        count jobs
        2     gzip
        2     trim_fastq
        4
rule trim_fastq:
    input: b.fastq
    output: b.trimmed.fastq
    jobid: 2
    wildcards: prefix=b
Finished job 2.
1 of 4 steps (25%) done
rule gzip:
    input: b.trimmed.fastq
    output: b.trimmed.fastq.gz
    jobid: 0
    wildcards: prefix=b.trimmed.fastq
Removing temporary output file b.trimmed.fastq.
Finished job 0.
2 of 4 steps (50%) done
rule trim_fastq:
    input: a.fastq
    output: a.trimmed.fastq
    jobid: 3
    wildcards: prefix=a
Finished job 3.
3 of 4 steps (75%) done
rule gzip:
    input: a.trimmed.fastq
    output: a.trimmed.fastq.gz
    jobid: 1
    wildcards: prefix=a.trimmed.fastq
Removing temporary output file a.trimmed.fastq.
Finished job 1.
4 of 4 steps (100%) done
```

Snakemake

```
rule trim_fastq:
    input: "{prefix}.fastq"
    output: temp("{prefix}.trimmed.fastq")
    shell:
        "trim_galore --illumina {input} > {output}"

rule gzip:
    input: "{prefix}"
    output: "{prefix}.gz"
    shell:
        "gzip -c {input} > {output}"
```

What happens if we add another sample?

```
$ snakemake {a,b}.trimmed.fastq.gz
Building DAG of jobs...
Using shell: /bin/bash
Provided cores: 1
Rules claiming more threads will be scaled down.
Job counts:
        count jobs
        2     gzip
        2     trim_fastq
        4
rule trim_fastq:
    input: b.fastq
    output: b.trimmed.fastq
    jobid: 2
    wildcards: prefix=b
Finished job 2.
1 of 4 steps (25%) done
rule gzip:
    input: b.trimmed.fastq
    output: b.trimmed.fastq.gz
    jobid: 0
    wildcards: prefix=b.trimmed.fastq
Removing temporary output file b.trimmed.fastq.
Finished job 0.
2 of 4 steps (50%) done
rule trim_fastq:
    input: a.fastq
    output: a.trimmed.fastq
    jobid: 3
    wildcards: prefix=a
Finished job 3.
3 of 4 steps (75%) done
rule gzip:
    input: a.trimmed.fastq
    output: a.trimmed.fastq.gz
    jobid: 1
    wildcards: prefix=a.trimmed.fastq
Removing temporary output file a.trimmed.fastq.
Finished job 1.
4 of 4 steps (100%) done
```

Snakemake

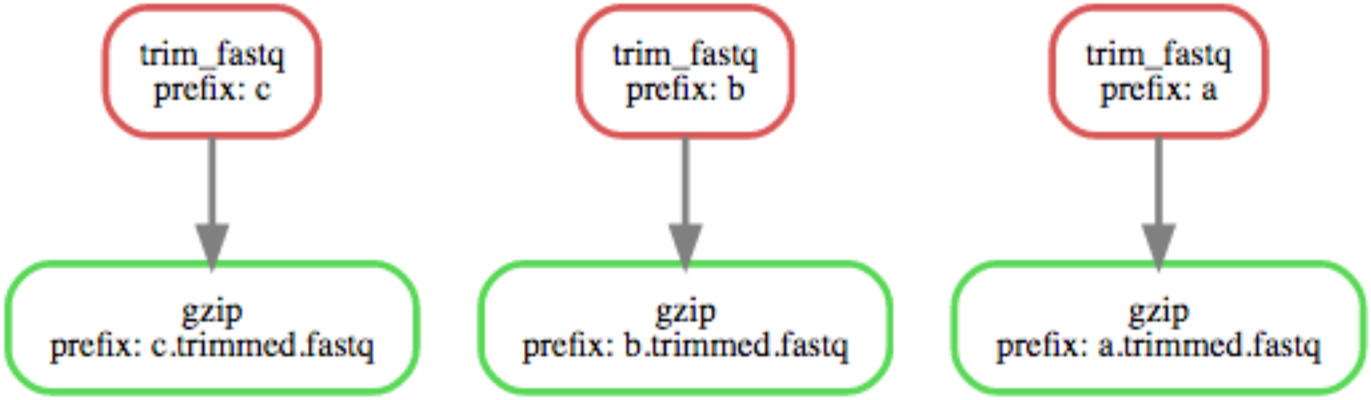
```
rule trim_fastq:
    input: "{prefix}.fastq"
    output: temp("{prefix}.trimmed.fastq")
    shell:
        "trim_galore --illumina {input} > {output}"

rule gzip:
    input: "{prefix}"
    output: "{prefix}.gz"
    shell:
        "gzip -c {input} > {output}"
```

```
$ snakemake {a,b,c}.trimmed.fastq.gz
Building DAG of jobs...
Using shell: /bin/bash
Provided cores: 1
Rules claiming more threads will be scaled down.
Job counts:
        count jobs
        1     gzip
        1     trim_fastq
        2

[Thu Aug 15 15:29:56 2019]
rule trim_fastq:
    input: c.fastq
    output: c.trimmed.fastq
    jobid: 5
    wildcards: prefix=c

[Thu Aug 15 15:29:56 2019]
Finished job 5.
1 of 2 steps (50%) done

[Thu Aug 15 15:29:56 2019]
rule gzip:
    input: c.trimmed.fastq
    output: c.trimmed.fastq.gz
    jobid: 2
    wildcards: prefix=c.trimmed.fastq

Removing temporary output file c.trimmed.fastq.
[Thu Aug 15 15:29:56 2019]
Finished job 2.
2 of 2 steps (100%) done
Complete log:
/Users/verenakutschera/nobackup/misc/teaching/2019
0903_snakemake_intro_EBC/workflows/.snakemake/log/
2019-08-15T152956.351207.snakemake.log
```

Rule graphs

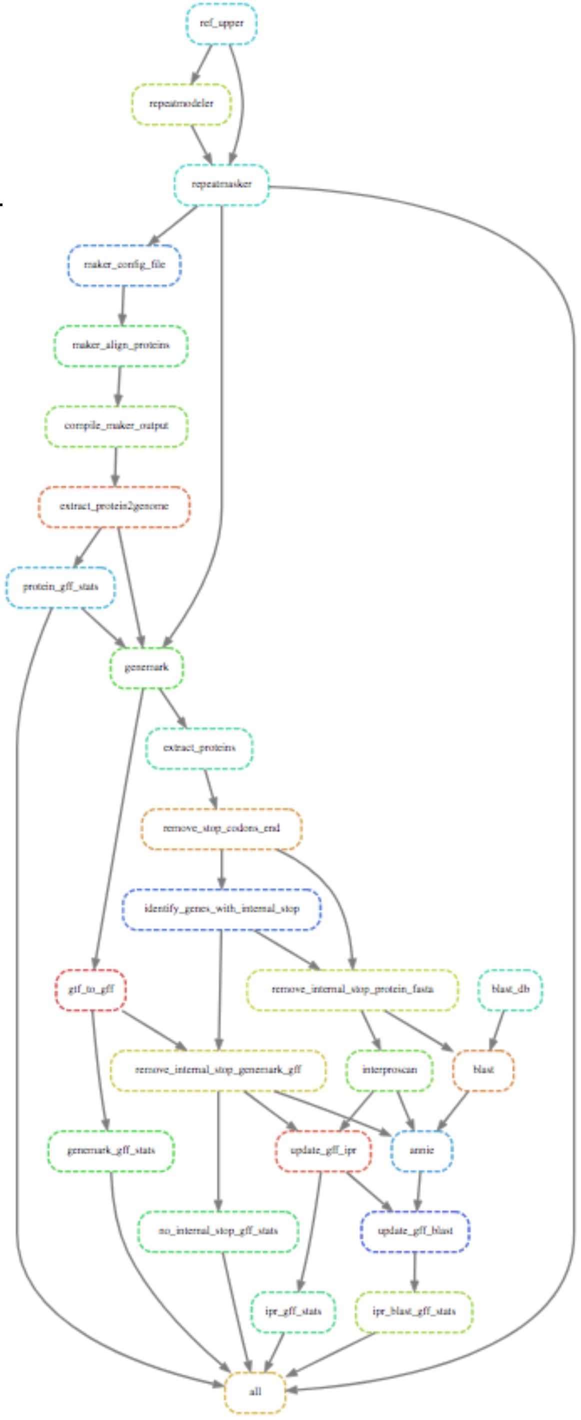
```
rule trim_fastq:
    input: "{prefix}.fastq"
    output: temp("{prefix}.trimmed.fastq")
    shell:
        "trim_galore --illumina {input} > {output}"

rule gzip:
    input: "{prefix}"
    output: "{prefix}.gz”
    shell:
        "gzip -c {input} > {output}”
```

- Snakemake keeps track of when files were generated and by which rules



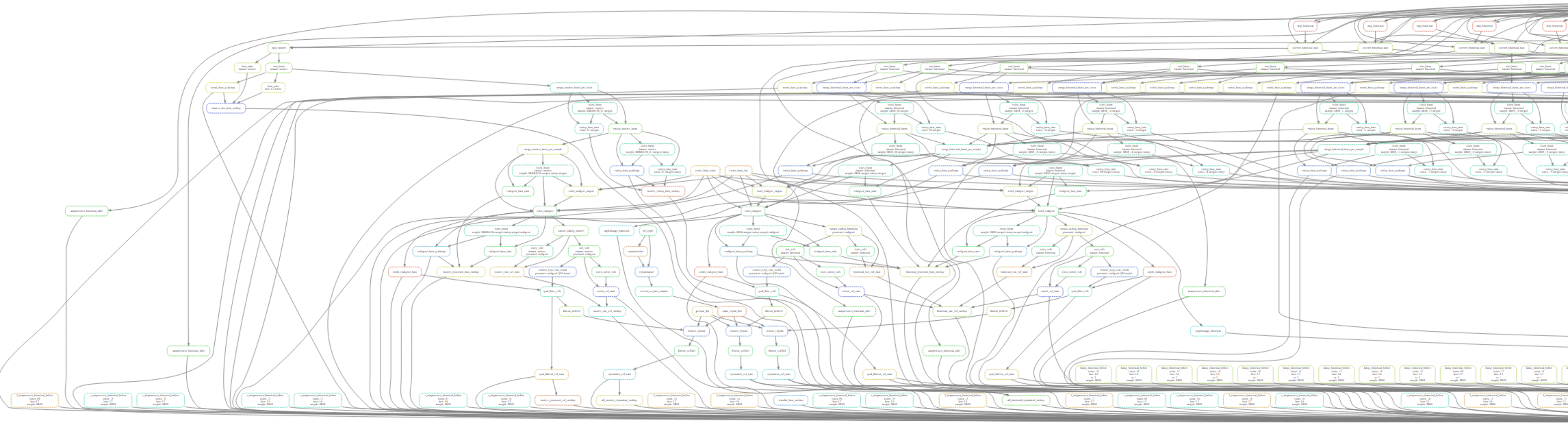
Rule graphs

- Workflows can get complex...



Rule graphs

- Workflows can get complex...



Snakemake advantages

- Structured pipeline
- Ideal for parallel problems (i.e. analysis of many samples)
- Dependencies are handled
- Execution order is inferred from input and output file names
- Reentrance into analysis is possible
- New upstream files are automatically analyzed
- Logging system to follow the status

Snakemake advantages

- Easy handling on a cluster
- Integration of various code, e.g. R, bash and python
- Rule syntax very flexible and adaptable

- Python
  - Glue language
  - Snakemake is written in python
  - Snakemake syntax is inspired by python

- Active development and large community

Snakemake disadvantages

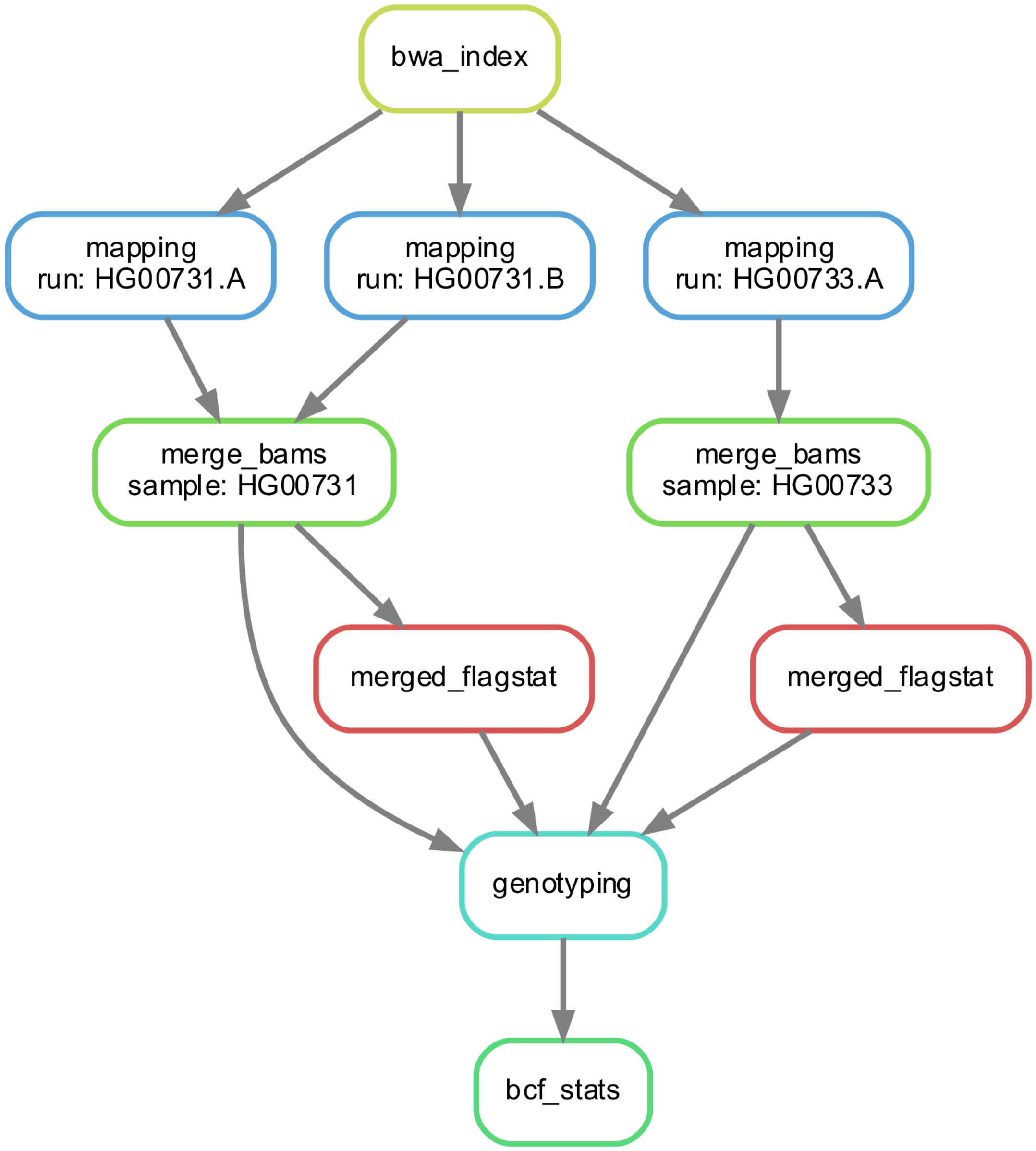
- Snakefiles are no python modules
- Errors sometimes cryptic and difficult to track down
- Risk to end up in an infinite loop or to delete the content of a file
  - Do a lot of testing
  - Backup the data
- Rule syntax is very flexible and adaptable
  - Different developers will come up with different solutions for the same problem – difficult to share workflows

Demo

- Convert bash script to snakemake workflow
- Run the workflow on UPPMAX
- Final workflow will be on data science GitHub repository (along with this presentation)

Demo

- FastQ files from two samples
  - HG00731 (run A and B)
  - HG00733 (run A)
- Map to reference genome
- Merge BAM files per sample
- Get BAM file statistics
- Joint genotyping
- Get VCF file statistics



Demo I

- Snakefile
- Basic rule syntax
- Python functions in snakemake workflows
- Input lists
- “Expand” statement
- Dry mode
- Rule graphs (dag files)
- Target rule (“all”)
- Temporary files

How to run snakemake on UPPMAX

- Tmux or screen to send snakemake to the background
  - tmux: check e.g. https://danielmiessler.com/study/tmux/
  - Write the standard output to a file with “&>” if you are running a long workflow or many samples at once

How to run snakemake on UPPMAX

- Snakemake with cluster configuration sends each rule as job to the slurm system
  - cluster.json or cluster.yaml with slurm parameters for each rule
    - To use yaml on Rackham, install the python package PyYAML for python version 3.6.0 or higher (https://www.uppmax.uu.se/support/user-guides/python-modules-guide/)
  - Start snakemake with this command to point to the cluster.yaml file:

```
$ snakemake -j 999 --cluster-config cluster.yaml \
--cluster "sbatch -A {cluster.account} \
-p {cluster.partition} -n {cluster.n} -t {cluster.time}"
```

Typical issues on the slurm system

- How do I cancel snakemake when it’s running on the slurm system?
  1. cancel jobs (scancel)
  2. cancel snakemake process in tmux/screen session with CTRL+C

Typical issues on the slurm system

- How do I cancel snakemake when it's running on the slurm system?
  1. cancel jobs (scancel)
  2. cancel snakemake process in tmux/screen session with CTRL+C

- What do I do if a specific rule/job failed?
  – `"Error in rule XYZ"` in standard output
  – Look for the rule and the line `"cluster_jobid: Submitted batch job 1234567"` in standard output
  – Slurm output `"slurm-1234567.out"` will hopefully contain more info

Typical issues on the slurm system

- How do I cancel snakemake when it’s running on the slurm system?
  1. cancel jobs (scancel)
  2. cancel snakemake process in tmux/screen session with CTRL+C

- What do I do if a specific rule/job failed?
  - `"Error in rule XYZ”` in standard output
  - look for the rule and the line `"cluster_jobid: Submitted batch job 1234567”` in standard output
  - slurm output `“slurm-1234567.out”` will hopefully contain more info

- The jobs submitted by snakemake are gone from the list of running jobs, but snakemake is still running. What happened, and how do I fix it?
  - snakemake and slurm are communicating about running jobs, except for jobs failing due to timeout
  - cancel the snakemake process with CTRL+C and restart workflow

Demo II

- Threads
- Cluster configuration

How to make UPPMAX happy

- Snakemake sends many, small jobs to the slurm system
  - Define group for execution to send several rules in one job to slurm

```
samples = [1,2,3,4,5]

rule a:
    input:
        "a/{sample}.out"
    output:
        "a/{sample}.edits.out"
    group: "mygroup"
    shell:
        "touch {output}"

rule b:
    input:
        "a/{sample}.edits.out"
    output:
        "b/{sample}.out"
    group: "mygroup"
    shell:
        "touch {output}"

rule c:
    input:
        expand("b/{sample}.out", sample=samples)
    output:
        "test.out"
    shell:
        "touch {output}"
```

- Rules a and b are run as one job per sample in the group “mygroup”
- Rule c connects the other jobs
  - As part of “mygroup”, everything would be submitted as one group

How to make UPPMAX happy

- Snakemake sends many, small jobs to the slurm system
  - Define group for execution to send several rules in one job to slurm
  - Define local rules

```
localrules: all, foo

rule all:
  input: ...

rule foo:
...

rule bar:
...
```

- Rules all and foo will be executed on the login node
- Rule foo will be submitted as a job

Demo III

- Defining groups for execution
- Local rules
- Config files
- Non-file parameters for rules
- Running python code instead of a shell command
- Log files

Snakemake help

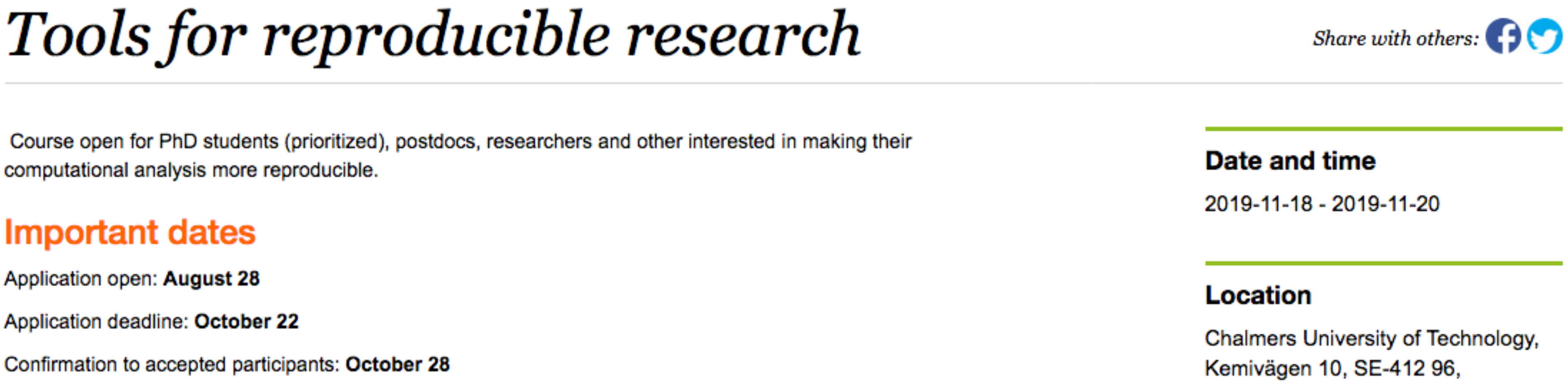
- Snakemake documentation on readthedocs:
  - https://snakemake.readthedocs.io/en/stable/#
- Stackoverflow:
  - https://stackoverflow.com/questions/tagged/snakemake
- NBIS bioinformatics drop-in:
  - https://nbis.se/events/
  - Uppsala: every Thursday 10-11h in Navet, BMC Uppsala E10:3

Snakemake tutorials

- Snakemake documentation:
  - https://snakemake.readthedocs.io/en/stable/tutorial/tutorial.html
- NBIS reproducible research workshop:
  - https://nbis-reproducible-research.readthedocs.io/en/latest/snakemake/

Snakemake tutorials

- NBIS reproducible research workshop:
  - https://nbis-reproducible-research.readthedocs.io/en/latest/snakemake/



Tools for reproducible research

Share with others:

Course open for PhD students (prioritized), postdocs, researchers and other interested in making their computational analysis more reproducible.

Important dates

Application open: **August 28**

Application deadline: **October 22**

Confirmation to accepted participants: **October 28**

Date and time

2019-11-18 - 2019-11-20

Location

Chalmers University of Technology, Kemivägen 10, SE-412 96,

- IN THE MAKING: 2-day NBIS snakemake workshop, fall 2020

Example workflows as inspiration

- RNA-seq:
  - https://github.com/crazyhottommy/RNA-seq-analysis/tree/master/RNA-seq-snakemake-pipeline
  - https://github.com/slowkow/snakefiles
- GATK variant calling:
  - https://github.com/snakemake-workflows/dna-seq-gatk-variant-calling
- Samtools variant calling (incl. Docker / Singularity):
  - https://github.com/fbartusch/snakemake_tutorial
- ACCEL amplicon trimming:
  - https://github.com/snakemake-workflows/accel-amplicon-trimming
- Single-cell drop-seq:
  - https://github.com/snakemake-workflows/single-cell-drop-seq

References & sources for slides

- Presentations by NBIS colleagues Leif Wigge & Rasmus Ågren, Per Unneberg
- Snakemake demo inspired by material from Per Unneberg
- https://groupes.renater.fr/sympa/d_read/bios4biol/2017/VisioCati2017/slides_snakemake.pdf
- Snakemake tutorial & documentation: https://snakemake.readthedocs.io/en/stable/index.html