ETH zürich

Snakemake introduction

Scientific IT Services
Michal Okoniewski

What is Snakemake?

What is Snakemake?

- Snakemake is a **workflow management system** for creation and execution of data analysis workflows. It is popular in the field of bioinformatics, where complex data processing and analysis pipelines are common.

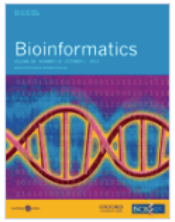
- At its core, Snakemake is a **Python-based** workflow language that allows you to define rules, dependencies, and inputs/outputs for each step of a workflow. These rules are written in a human-readable text file called a Snakefile.

Snakemake initial paper

Bioinformatics

ISCB INTERNATIONAL SOCIETY FOR COMPUTATIONAL BIOLOGY

Issues | Advance articles | Submit | Alerts | About | Bioinformatics | Advanced Search



Volume 28, Issue 19
October 2012

Article Contents

< Previous Next >

JOURNAL ARTICLE

Snakemake—a scalable bioinformatics workflow engine FREE

Johannes Köster ✉, Sven Rahmann Author Notes

Bioinformatics, Volume 28, Issue 19, October 2012, Pages 2520–2522,
https://doi.org/10.1093/bioinformatics/bts480
Published: 20 August 2012 **Article history**

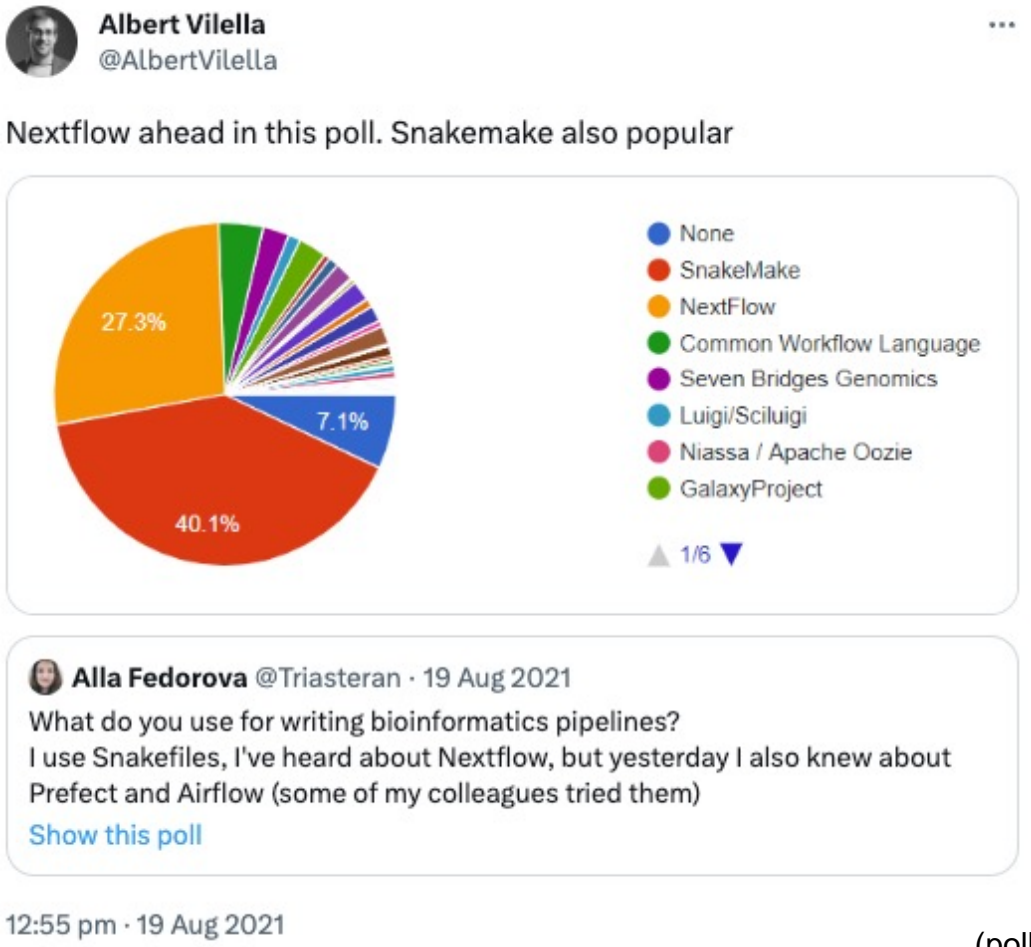
A correction has been published: *Bioinformatics*, Volume 34, Issue 20, October 2018, Page 3600, https://doi.org/10.1093/bioinformatics/bty350

PDF | Split View | Cite | Permissions | Share

Abstract

Summary: Snakemake is a workflow engine that provides a readable Python-based workflow definition language and a powerful execution environment that scales from single-core workstations to compute clusters without modifying the workflow. It is the first system to support the use of automatically inferred multiple named wildcards (or variables) in input and output filenames.

Snakemake is popular among bioinformaticians

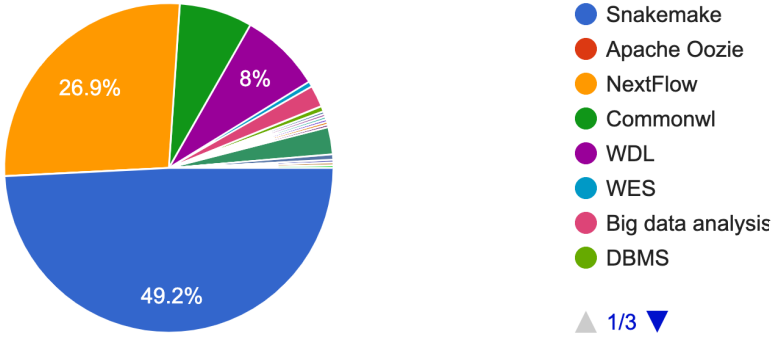


(poll among 550 bioinformaticians)

Snakemake is popular among bioinformaticians

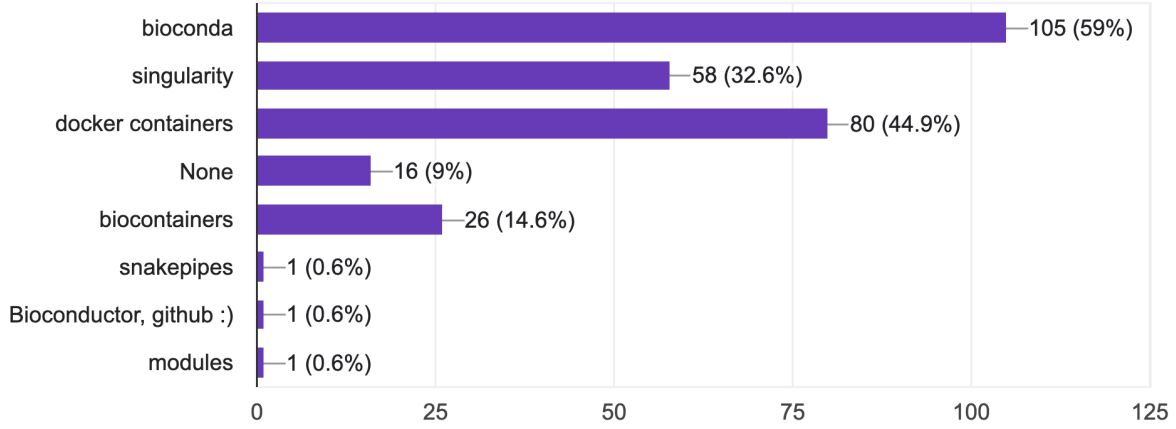
Which #Bioinformatics #Languages #Specifications #Standards do you use or prefer? bit.ly/biowl

376 responses



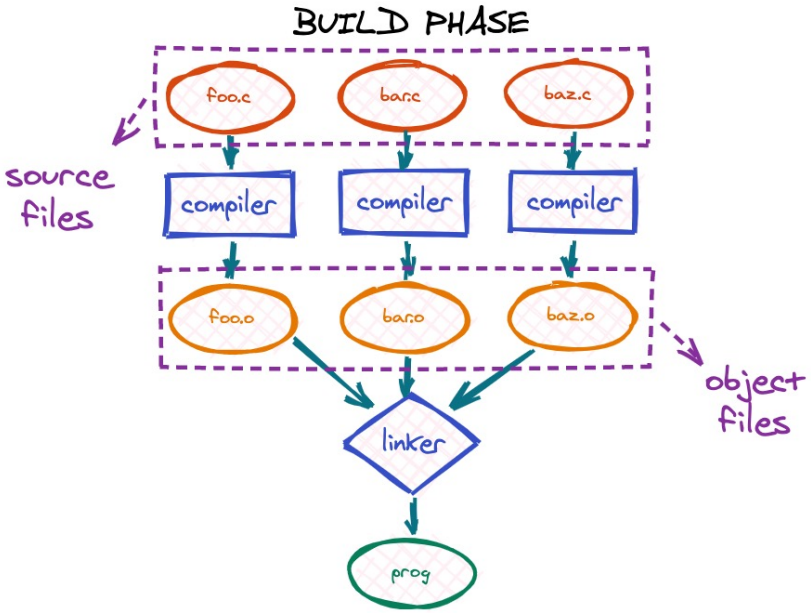
Which of the following Technologies / Communities / Frameworks do you prefer? bit.ly/biowl

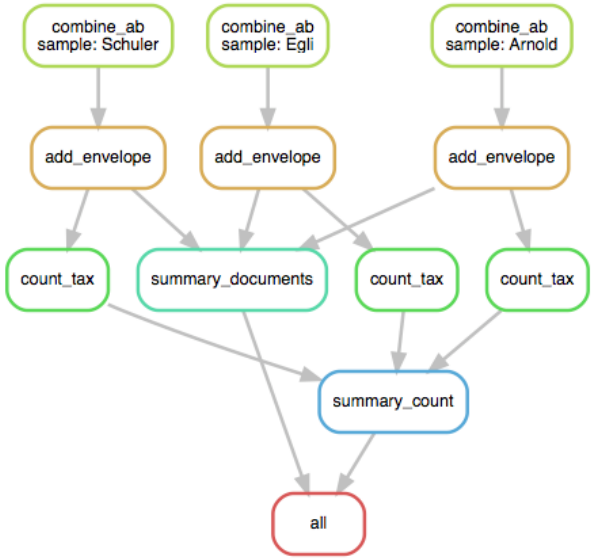
178 responses



Make-like principles

- Snakemake automatically analyzes the dependencies between rules and efficiently executes them in the correct order, taking advantage of parallelization and resource management.





Make-like principles

cmake

```
# Compiler and flags
CC = gcc
CFLAGS = -Wall -Wextra

# Directories
SRC_DIR = src
BUILD_DIR = build
BIN_DIR = bin

# Source files
SRCS = $(wildcard $(SRC_DIR)/*.c)

# Object files
OBJS = $(patsubst $(SRC_DIR)/%.c,$(BUILD_DIR)/%.o,$(SRCS))

# Binary
TARGET = $(BIN_DIR)/myprogram

# Default target
all: $(TARGET)

# Compile object files
$(BUILD_DIR)/%.o: $(SRC_DIR)/%.c
            @mkdir -p $(BUILD_DIR)
            $(CC) $(CFLAGS) -c $< -o $@

# Build the binary
$(TARGET): $(OBJS)
            @mkdir -p $(BIN_DIR)
            $(CC) $(CFLAGS) $^ -o $@

# Clean build files
clean:
            rm -rf $(BUILD_DIR) $(BIN_DIR)
```

Snakemake

```
SAMPLES, = glob_wildcards("data/{sample}_a.txt")

rule all:
    input:
        "reports/final_marker.txt"

rule combine_ab:
    input:
        a="data/{sample}_a.txt",
        b="data/{sample}_b.txt"
    output:
        "data_processed/{sample}.txt"
    shell:
        "cat {input.a} {input.b} > {output}"

rule add_envelope:
    input:
        combined="data_processed/{sample}.txt",
        envelope="data/{sample}_envelope.txt"
    output:
        "reports/{sample}.txt"
    shell:
        "cat {input.envelope} {input.combined} {input.envelope} > {output}"

rule summary_all:
    input:
        expand("reports/{sample}.txt", sample=SAMPLES)
    output:
        "reports/final_marker.txt"
    shell:
        "echo {input} \n"
        "touch reports/final_marker.txt"
```

Key benefits of Snakemake

- Modularization of data processing
- Re-using partial results in case of crashes or errors in complex workflows
- Reproducibility
  - Same results for the same data, eg repeating what presented in a paper
  - Corresponding processing for data integration, eg integrating data from public repositories
- Scalability and portability of data processing workflows
  - Laptop/HPC/cloud
- Intergation with Conda, Bioconda and containers

Installation and configuration

- Recommended way is conda/mamba in a separate environment

```
$ mamba create -c conda-forge -c bioconda -n snakemake snakemake
$ conda activate snakemake
$ snakemake --help
```

- On Euler, a recent version is in a python module
  - `module load gcc/8.2.0 python/3.11.2`
- Pip installation possible
  - but not recommended by the authors, as “Snakemake has non-python dependencies”

Workflow structure

Workflow structure

- The main part of the workflow is **Snakefile**.
- Snakefile consists of **freeform python and rules**
- Rules include at least the specification of **input, output and processing**
- There is a special rule, “**rule all**”, with the final output
- **Wildcards**: placeholders that represent parts of filenames or patterns. Used for dynamic file generation and flexible rule matching. Wildcards can be used in the input and output within rules.
- The workflow is accompanied with **configuration**
- There may be **additional infrastructure** of the workflow such as containers, logging, templates, …

Workflow structure – freeform python header

```
import glob                                                    # python imports
import os

shellcommand="mkdir data"                                      #preparation of folders and data structures
os.system(shellcommand)
shellcommand="mkdir data_processed"
os.system(shellcommand)
shellcommand="mkdir reports"
os.system(shellcommand)


SAMPLES, = glob_wildcards("data/{sample}_a.txt")               # filling the wildcards, creating dictionaries
print("=====")                                                 # debugging
print(SAMPLES)
print("=====")
```

Workflow structure – wildcards

```
# creating wildcards
import glob
SAMPLES, = glob_wildcards("data/{sample}.fastq.gz")
print(SAMPLES)

# simple use
rule samtools_index:
    input:
        "sorted_reads/{sample}.bam"
    output:
        "sorted_reads/{sample}.bam.bai"
    shell:
        "module load samtools \n"
        "samtools index {input}"
```

Workflow structure – wildcards

```
# multiple wildcards use

rule complex_conversion:
    input:
        "{dataset}/inputfile"
    output:
        "{dataset}/file.{group}.txt"
    shell:
        "somecommand --group {wildcards.group} < {input} > {output}"
```

Workflow structure – rules

```
localrules: all, summary_all                    # rules executed locally, not on the compute
                                                # nodes, e.g. control ones or simple I/O

rule all:
    input:                                      # the final product
        "reports/final_marker.txt"

rule combine_ab:                                # rule
    input:                                      # input with {sample} wildcard
        a="data/{sample}_a.txt",
        b="data/{sample}_b.txt"
    output:                                     # output with {sample} wildcard
        "data_processed/{sample}.txt"
    shell:                                      # processing with shell for bash or run for python
        "cat {input.a} {input.b} > {output}"
```

Rule implementation – a simple rule

```
rule combine_ab:
    input:
        a="data/{sample}_a.txt",
        b="data/{sample}_b.txt"
    output:
        "data_processed/{sample}.txt"
    shell:
        "cat {input.a} {input.b} > {output}"
```

Rule implementation – a simple rule with multiple outputs

```
rule process_sequences:
    input:
        "input_file.fasta"
    output:
        reverse_complement="output_files/{sample}.reverse_complement.fasta",
        uppercase="output_files/{sample}.uppercase.fasta"
    shell:
        """
        python process_sequences.py {input} \
            reverse_complement={output.reverse_complement} \
            uppercase={output.uppercase}
        """
```

Rule implementation – multiple input “sink”

```
rule summary_all:
    input:
        expand("reports/{sample}.txt", sample=SAMPLES)
    output:
        "reports/all_reports.txt"
    shell:
        "echo {input} \n”
        "cat {input} > {output}"
```

Rule implementation – multiple input “sink”

```
rule create_count_table:
    input:
        expand("secondary_analysis/{sample}.cnt", sample = SAMPLES)
    output:
        "secondary_analysis/counts.csv"
    run:
        import pandas
        import glob

        filez = glob.glob('secondary_analysis/*.cnt')
        t1 = pandas.read_table(filez[1], header=1)
        tout = t1.iloc[:,0]
        for f in filez:
           t1=pandas.read_table(f, header=1)
           tout=pandas.concat([tout, t1.iloc[:,6]], axis=1)
           print(f)

        tout.to_csv('secondary_analysis/counts.csv')
```

Rule implementation – other parametrization types

```
rule align:
    input:
        "{sample}.fq",
        reference="ref.fa",
    output:
        "{sample}.sam"
    params:
        opts="-a -x map-ont",
    threads: 4
    log:
        "align/{sample}.log"
    conda:
        "envs/align.yaml"
    script:
        "scripts/align.sh"
```

```
#!/usr/bin/env bash

echo "Aligning sample ${snakemake_wildcards[sample]} with
minimap2" 2> "${snakemake_log[0]}"

minimap2 ${snakemake_params[opts]} -t ${snakemake[threads]}
"${snakemake_input[reference]}" \
    "${snakemake_input[0]}" > "${snakemake_output[0]}" 2>>
"${snakemake_log[0]}"
```

Workflow structure – parsing the workflow

- Starting from the input of “rule all”, the parser searches for output needed to produce it
- Rules producing this output are identified
- Then, iteratively, checks what output is needed to produce the input of those rules
- The inputs and outputs are matched, taken into consideration the content of wildcards
- This way, a directed acyclic graph (DAG) is produced.
- The DAG It leads from initial input, all the way to the “rule all”. This is how the execution of the workflow goes.

Workflow structure – parsing the workflow

Tell Snakemake what files you want to be created

```
rule:
  input: "A.txt", "B.txt", "C.txt"
```

Produce the files you want to have from some intermediate result

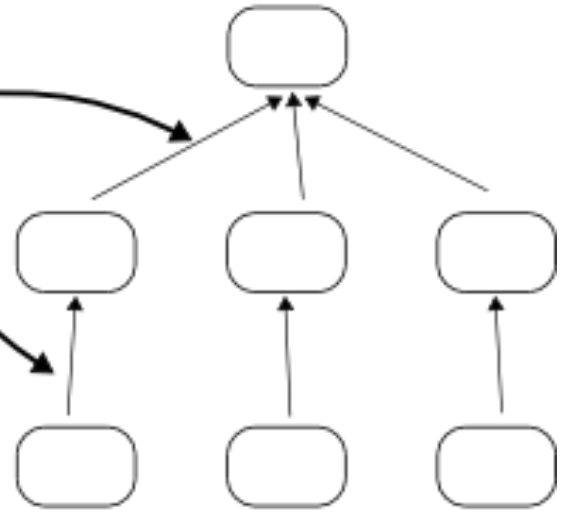
```
rule:
  input: "{sample}.inter"
  output: "{sample}.txt"
  shell: "somecommand {input} {output}"
```

Create a needed intermediate result

```
rule:
  input:  "{sample}.in"
  output: "{sample}.inter"
  run:
    somepythoncode()
```

Use wildcards to write general rules for all samples

Snakemake determines the dependencies for you



(from Snakemake documentation)

Rule configuration

- One rule => one job on the cluster
- For every wildcard value
  - or for any combination of multiple wildcards
- With input and output according to DAG
- With exection parameters according to the configuration
  - sent as a job to the compute nodes
    - except „localrules”

Rule configuration – old style, cluster.json

```
{
    "__default__" :
    {
        "time" : "03:55:00",
        "n" : 24,
        "mem" : 2560
    },
    "gatk_prep" :
    {
        "n" : 1,
        "mem" : 16384
    }
}
```

Rule configuration – using a profile

- Template of cluster run profiles can be found on github
  - https://github.com/snakemake-profiles/doc
  - https://github.com/jdblischak/smk-simple-slurm

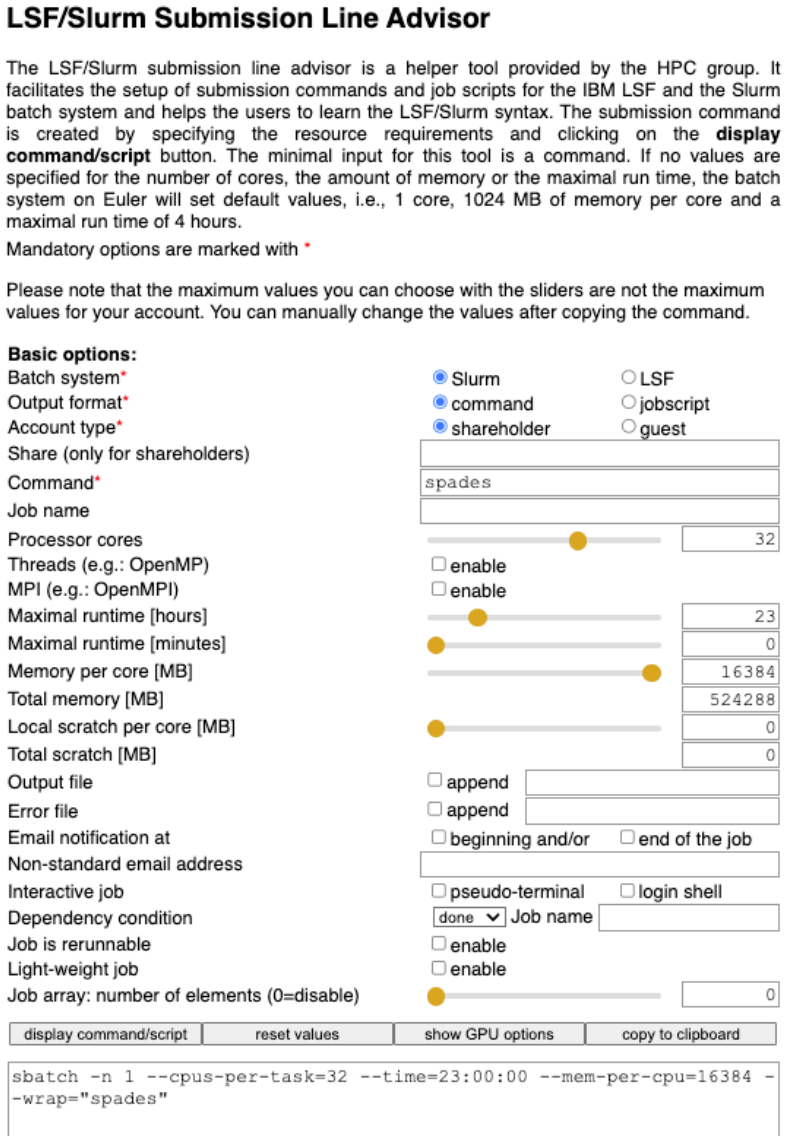
Rule configuration – using a profile, config.yaml

```
#config.yaml

cluster:
  mkdir -p logs/{rule} &&
  sbatch
    --partition={resources.partition}
    --qos={resources.qos}
    --cpus-per-task={threads}
    --mem={resources.mem_mb}
    --job-name=smk-{rule}-{wildcards}
    --output=logs/{rule}/{rule}-{wildcards}-
%j.out
    --time={resources.time}
default-resources:
  - partition=<name-of-default-partition>
  - qos=<name-of-quality-of-service>
  - mem_mb=1000
  - time="01:00:00"
```

```
#configuration in the rule

  rule more_time_and_memory:
    resources:
        time = "03:00:00”
        mem_mb=64000
```

Rule configuration help

https://scicomp.ethz.ch/public/lsla/index2.html



LSF/Slurm Submission Line Advisor

The LSF/Slurm submission line advisor is a helper tool provided by the HPC group. It facilitates the setup of submission commands and job scripts for the IBM LSF and the Slurm batch system and helps the users to learn the LSF/Slurm syntax. The submission command is created by specifying the resource requirements and clicking on the **display command/script** button. The minimal input for this tool is a command. If no values are specified for the number of cores, the amount of memory or the maximal run time, the batch system on Euler will set default values, i.e., 1 core, 1024 MB of memory per core and a maximal run time of 4 hours.

Mandatory options are marked with *

Please note that the maximum values you can choose with the sliders are not the maximum values for your account. You can manually change the values after copying the command.

Basic options:

Batch system*	Slurm / LSF
Output format*	command / jobscript
Account type*	shareholder / guest
Share (only for shareholders)	
Command*	spades
Job name	
Processor cores	32
Threads (e.g.: OpenMP)	enable
MPI (e.g.: OpenMPI)	enable
Maximal runtime [hours]	23
Maximal runtime [minutes]	0
Memory per core [MB]	16384
Total memory [MB]	524288
Local scratch per core [MB]	0
Total scratch [MB]	0
Output file	append
Error file	append
Email notification at	beginning and/or / end of the job
Non-standard email address	
Interactive job	pseudo-terminal / login shell
Dependency condition	done, Job name
Job is rerunnable	enable
Light-weight job	enable
Job array: number of elements (0=disable)	0

display command/script | reset values | show GPU options | copy to clipboard

```
sbatch -n 1 --cpus-per-task=32 --time=23:00:00 --mem-per-cpu=16384 -
-wrap="spades"
```

Using various software stacks

Using various software stacks

The software used by Snakemake may be eg.

- Locally installed
- In a rule-specific conda environment
- From a module on the cluster
- Using a container, eg Singularity one

Software stacks - locally installed, “laptop mode”

```
rule trim:
    input:
        "data/{sample}.fastq.gz"
    output:
        "trimmed_data/{sample}.fastq.gz"
    run:
        shell(
        'java -jar trimmomatic-0.39.jar SE -phred33 {input} {output}
ILLUMINACLIP:'+TRIMFILE+':2:30:10 LEADING:3 TRAILING:3 SLIDINGWINDOW:4:15 MINLEN:36'
        )
```

Software stacks - modules on the cluster

```
rule trim:
    input:
        "data/{sample}.fastq.gz"
    output:
        "trimmed_data/{sample}.fastq.gz"
    run:
        shell(
        'module load trimmomatic \n'+
        'trimmomatic SE -phred33 {input} {output} ILLUMINACLIP:'+TRIMFILE+':2:30:10 LEADING:3
TRAILING:3 SLIDINGWINDOW:4:15 MINLEN:36'
        )

# running
snakemake -p -j 999 --cluster-config cluster.json --cluster "sbatch --time {cluster.time} -n 1
--cpus-per-task={cluster.n} --mem-per-cpu={cluster.mem}"
```

Software stacks - Singularity containers

```
rule trim:
    input:
        "data/{sample}.fastq.gz"
    output:
        "trimmed_data/{sample}.fastq.gz"
    singularity:
        "https://depot.galaxyproject.org/singularity/trimmomatic:0.39--1"
    shell:
        "trimmomatic SE -phred33 {input} {output} ILLUMINACLIP:"+TRIMFILE+":2:30:10 LEADING:3
TRAILING:3 SLIDINGWINDOW:4:15 MINLEN:36”

# running
snakemake -p -j 999 --use-singularity --cluster-config cluster.json \
--cluster "sbatch --time {cluster.time} -n 1 --cpus-per-task={cluster.n}" \
--singularity-args "--bind /cluster/scratch/myuser/test1/:/mnt2 --bind /cluster/home/ myuser/
hisat/grch38/:/genomes --bind bind /cluster/home/ myuser/ hisat /hg38/:/annots"
```

Software stacks - Singularity containers from Galaxy Depot

- 85’000 containers for bioinformatic software
  - various versions
  - Optimized for size, runnable state, faster download
  - https://depot.galaxyproject.org/singularity/
- Snakemake does caching the containers in the workflow internals (.snakemake)
  - Downloaded only once
- How to use in a single job or in Snakemake
  - https://scicomp.ethz.ch/wiki/Galaxy_Depot_Software_Stack

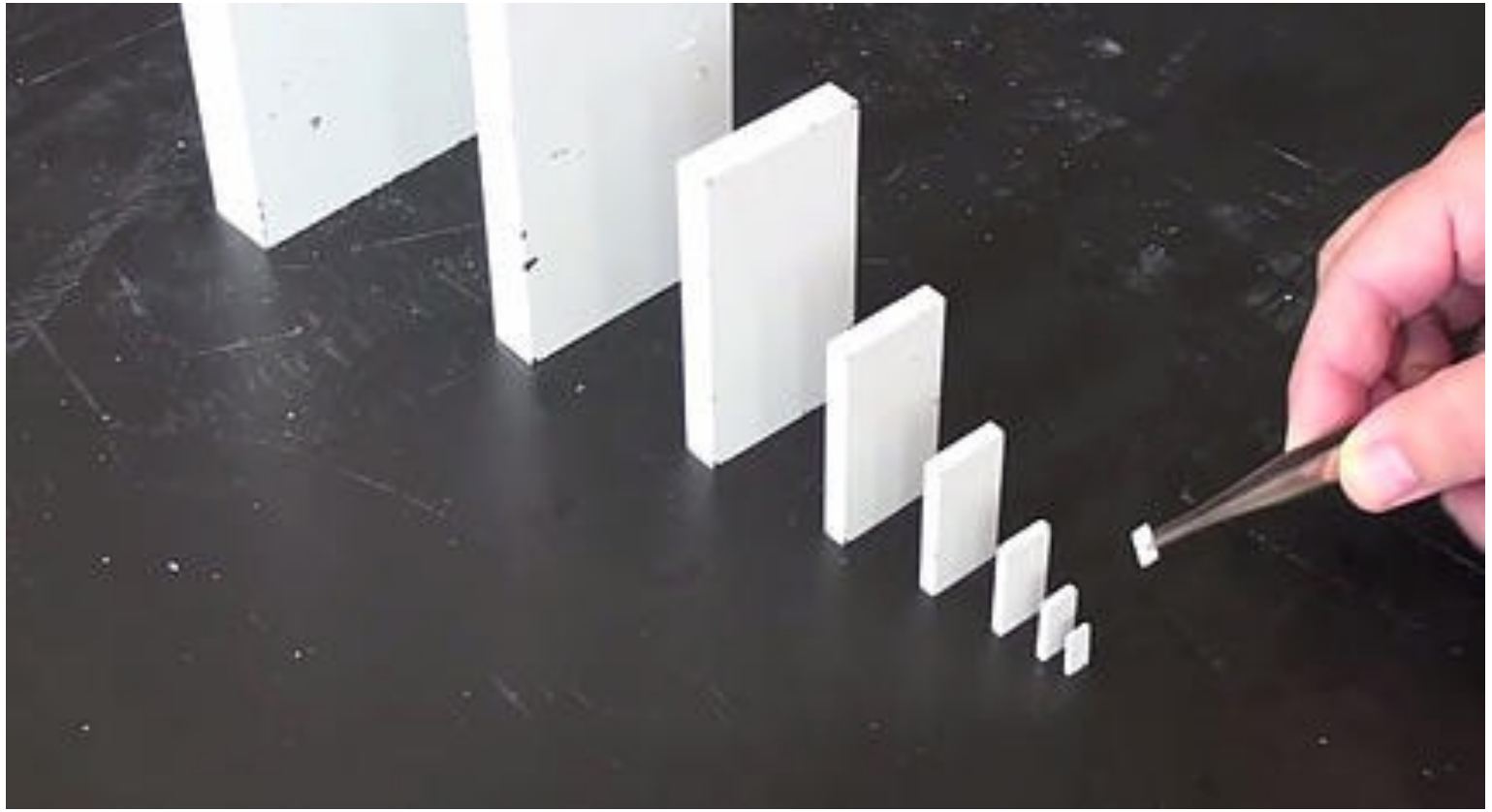
Software stacks - using a specific conda environment

```
# prepare the yaml
conda env export -n trimming > trimming-env.yaml

# in Snakefile
rule trimming_conda_env_rule:
    conda: " trimming-env.yaml"
    shell:
        "which cutadapt”

# running
snakemake --use-conda trimming_conda_env_rule

# conda dry run, creating the environments
snakemake --use-conda --conda-create-envs-only
```

Advanced workflow features

- Dynamic file generation
- Functions as input
- Sub-workflows and includes
- Wrappers
  - List of wrappers https://snakemake-wrappers.readthedocs.io/en/stable/wrappers.html
- Rule patterns and templating
- Data-dependent conditional execution and branching
- Common-Workflow-Language support
- …

Workflow execution



Workflow execution – dry run

- Option -n does not send the rules for execution
- Option -p prints the full output to the screen

- snakemake -np builds the DAG of rules and executes global python part

- A simple example for the practical tests
  - https://github.com/michalogit/snakemaketax

Workflow execution – dry run

```
(myenv) Michals-MacBook-Pro-2:snakemaketax michalo$ snakemake -np
mkdir: data: File exists
=====
['Schuler', 'Egli', 'Arnold']
=====

rule combine_ab:
    input: data/Egli_a.txt, data/Egli_b.txt
    output: data_processed/Egli.txt
    jobid: 6
    wildcards: sample=Egli

cat data/Egli_a.txt data/Egli_b.txt > data_processed/Egli.txt
(...)
Job counts:
count jobs
3 add_envelope
1 all
3 combine_ab
1 summary_all
8
```

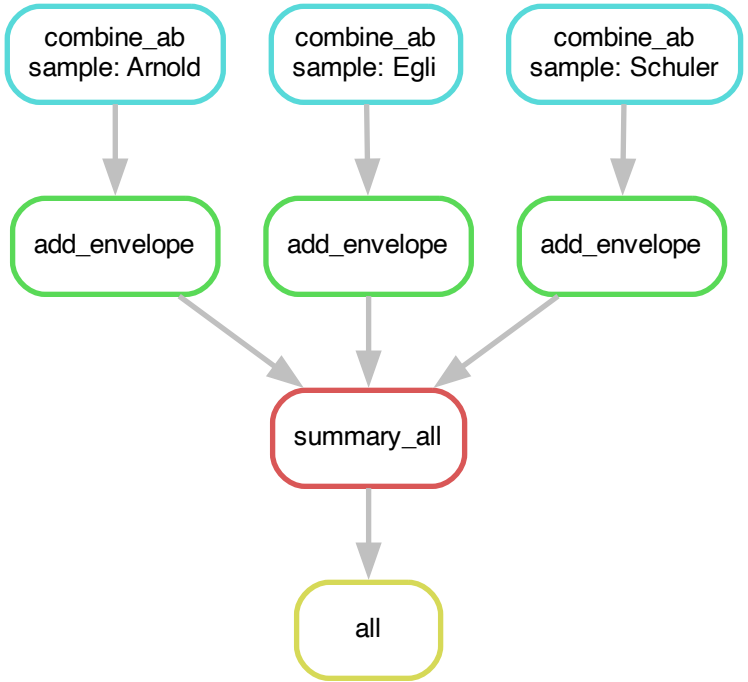
Workflow execution – graph check

```
(myenv) Michals-MacBook-Pro-2:snakemaketax michalo$ snakemake --dag
mkdir: data: File exists
=====
['Schuler', 'Egli', 'Arnold']
=====
digraph snakemake_dag {
    graph[bgcolor=white, margin=0];
    node[shape=box, style=rounded, fontname=sans,                 fontsize=10, penwidth=2];
    edge[penwidth=2, color=grey];
            0[label = "all", color = "0.17 0.6 0.85", style="rounded"];
            1[label = "summary_all", color = "0.00 0.6 0.85", style="rounded"];
            2[label = "add_envelope", color = "0.33 0.6 0.85", style="rounded"];
            3[label = "add_envelope", color = "0.33 0.6 0.85", style="rounded"];
            4[label = "add_envelope", color = "0.33 0.6 0.85", style="rounded"];
            5[label = "combine_ab\nsample: Egli", color = "0.50 0.6 0.85", style="rounded"];
            6[label = "combine_ab\nsample: Arnold", color = "0.50 0.6 0.85", style="rounded"];
            7[label = "combine_ab\nsample: Schuler", color = "0.50 0.6 0.85", style="rounded"];
            1 -> 0
            2 -> 1
            3 -> 1
            4 -> 1
            5 -> 2
            6 -> 3
            7 -> 4
}
```

Workflow execution – graph check

- The graph file needs to be edited to remove the python output
- The “dot” editor converts it to PDF

```
dot -Tpdf graph.dag > graph.pdf
```



Workflow execution – a proper run

- Local

```
snakemake -p
snakemake --cores 1
snakemake --snakefile Snakefile.my1
```

- Slurm

```
snakemake -p -j 999 --cluster-config cluster.json --cluster "sbatch --time {cluster.time} -n {cluster.n}
"
snakemake -p -j 999 --cluster-config cluster.json --cluster "sbatch --time {cluster.time} -n 1 --cpus-
per-task={cluster.n} "
snakemake -p -j 999 --cluster-config cluster.json --cluster "sbatch --time {cluster.time} -n 1 --cpus-
per-task={cluster.n} --mem-per-cpu={cluster.mem}"
```

- LSF

```
snakemake -p -j 999 --cluster-config cluster.json --cluster "bsub -W {cluster.time} -n {cluster.n}"
```

Workflow execution – dry run

```
(snakemake) [michalo@eu-login-05 Luca_dna]$ snakemake -np
mkdir: cannot create directory ‘data’: File exists
mkdir: cannot create directory ‘trimmed_data’: File exists
mkdir: cannot create directory ‘mapped_reads’: File exists
mkdir: cannot create directory ‘sorted_reads’: File exists
mkdir: cannot create directory ‘variants_gatk’: File exists
mkdir: cannot create directory ‘all_vcfs’: File exists
mkdir: cannot create directory ‘secondary_analysis’: File exists
['alphaV-genomicDNA_S1_L003', 'AlphaVBeta1-genomicDNA_S5_L003', 'WildTypeFibroblastWTF-genomicDNA_S6_L003', 'Beta3-genomicDNA_S3_L003',
'Beta1-genomicDNA_S2_L003', 'Beta5-genomicDNA_S4_L003']
Building DAG of jobs...
Job stats:
job                   count    min threads    max threads
-------------------  -------  -------------  -------------
all                       1              1              1
gatk_haplotypecaller      6              1              1
gatk_prep                 6              1              1
summary_gatk              1              1              1
total                    14              1              1


[Tue Jun 27 10:51:07 2023]
rule gatk_prep:
    input: sorted_reads/WildTypeFibroblastWTF-genomicDNA_S6_L003.bam, sorted_reads/WildTypeFibroblastWTF-genomicDNA_S6_L003.bam.bai
    output: variants_gatk/WildTypeFibroblastWTF-genomicDNA_S6_L003.dup.bam
    jobid: 38
    reason: Missing output files: variants_gatk/WildTypeFibroblastWTF-genomicDNA_S6_L003.dup.bam
    wildcards: sample=WildTypeFibroblastWTF-genomicDNA_S6_L003
    resources: tmpdir=/tmp

module load  gatk
module load samtools
gatk MarkDuplicates -I sorted_reads/WildTypeFibroblastWTF-genomicDNA_S6_L003.bam -O variants_gatk/WildTypeFibroblastWTF-genomicDNA_S6_L0
03.dup.bam -M output.metrics
samtools index variants_gatk/WildTypeFibroblastWTF-genomicDNA_S6_L003.dup.bam


[Tue Jun 27 10:51:07 2023]
rule gatk_prep:
    input: sorted_reads/Beta3-genomicDNA_S3_L003.bam, sorted_reads/Beta3-genomicDNA_S3_L003.bam.bai
    output: variants_gatk/Beta3-genomicDNA_S3_L003.dup.bam
    jobid: 40
    reason: Missing output files: variants_gatk/Beta3-genomicDNA_S3_L003.dup.bam
    wildcards: sample=Beta3-genomicDNA_S3_L003
```

Workflow execution – a proper run log

```
Building DAG of jobs...
Using shell: /cluster/apps/sfos/bin/bash
Provided cluster nodes: 999
Job stats:
job                count    min threads    max threads
---------------  -------  -------------  -------------
BWA                    6              1              1
all                    1              1              1
samtools_convert       6              1              1
samtools_index         6              1              1
samtools_sort          6              1              1
summary_bai            1              1              1
total                 26              1              1

Select jobs to execute...

[Thu Jun 22 13:43:31 2023]
rule BWA:
    input: trimmed_data/AlphaVBeta1-genomicDNA_S5_L003_R1.fastq.gz, trimmed_data/AlphaVBeta1-genomicDNA_S5_L003_R2.fastq.gz
    output: mapped_reads/AlphaVBeta1-genomicDNA_S5_L003.sam
    jobid: 10
    reason: Missing output files: mapped_reads/AlphaVBeta1-genomicDNA_S5_L003.sam
    wildcards: sample=AlphaVBeta1-genomicDNA_S5_L003
    resources: mem_mb=17881, mem_mib=17053, disk_mb=17881, disk_mib=17053, tmpdir=<TBD>

module load bwa/0.7.17
bwa mem -t 24 /cluster/home/michalo/project_michalo/mm39_bwa/mm39 trimmed_data/AlphaVBeta1-genomicDNA_S5_L003_R1.fastq.gz trimmed_data/AlphaVBeta1-
genomicDNA_S5_L003_R2.fastq.gz > mapped_reads/AlphaVBeta1-genomicDNA_S5_L003.sam
Submitted job 10 with external jobid 'Submitted batch job 19197619'.

[Thu Jun 22 13:43:31 2023]
rule BWA:
    input: trimmed_data/Beta3-genomicDNA_S3_L003_R1.fastq.gz, trimmed_data/Beta3-genomicDNA_S3_L003_R2.fastq.gz
    output: mapped_reads/Beta3-genomicDNA_S3_L003.sam
    jobid: 20
    reason: Missing output files: mapped_reads/Beta3-genomicDNA_S3_L003.sam
    wildcards: sample=Beta3-genomicDNA_S3_L003
    resources: mem_mb=22969, mem_mib=21905, disk_mb=22969, disk_mib=21905, tmpdir=<TBD>
```

Workflow execution – crash of a rule

```
module load bwa/0.7.17bwa mem -t 24 /cluster/home/michalo/project_michalo/mm39_bwa/mm39 trimmed_data/WildTypeFibroblastWTF-genomicDNA_S6_L003_R1.fastq.gz t
rimmed_data/WildTypeFibroblastWTF-genomicDNA_S6_L003_R2.fastq.gz > mapped_reads/WildTypeFibroblastWTF-genomicDNA_S6_L003.sam
Submitted job 25 with external jobid 'Submitted batch job 19120477'.
[Wed Jun 21 12:06:32 2023]
Error in rule BWA:
    jobid: 15
    input: trimmed_data/AlphaVBeta1-genomicDNA_S5_L003_R1.fastq.gz, trimmed_data/AlphaVBeta1-genomicDNA_S5_L003_R2.fastq.gz
    output: mapped_reads/AlphaVBeta1-genomicDNA_S5_L003.sam
    shell:
        module load bwa/0.7.17bwa mem -t 24 /cluster/home/michalo/project_michalo/mm39_bwa/mm39 trimmed_data/AlphaVBeta1-genomicDNA_S5_L003_R1.fastq.gz tri
mmed_data/AlphaVBeta1-genomicDNA_S5_L003_R2.fastq.gz > mapped_reads/AlphaVBeta1-genomicDNA_S5_L003.sam
        (one of the commands exited with non-zero exit code; note that snakemake uses bash strict mode!)
    cluster_jobid: Submitted batch job 19120425

Error executing rule BWA on cluster (jobid: 15, external: Submitted batch job 19120425, jobscript: /cluster/scratch/michalo/Luca_dna/.snakemake/tmp.3_v3ozd
f/snakejob.BWA.15.sh). For error details see the cluster log and the log files of the involved rule(s).
[Wed Jun 21 12:06:43 2023]
Error in rule BWA:
    jobid: 25
    input: trimmed_data/WildTypeFibroblastWTF-genomicDNA_S6_L003_R1.fastq.gz, trimmed_data/WildTypeFibroblastWTF-genomicDNA_S6_L003_R2.fastq.gz
    output: mapped_reads/WildTypeFibroblastWTF-genomicDNA_S6_L003.sam
    shell:
        module load bwa/0.7.17bwa mem -t 24 /cluster/home/michalo/project_michalo/mm39_bwa/mm39 trimmed_data/WildTypeFibroblastWTF-genomicDNA_S6_L003_R1.fa
stq.gz trimmed_data/WildTypeFibroblastWTF-genomicDNA_S6_L003_R2.fastq.gz > mapped_reads/WildTypeFibroblastWTF-genomicDNA_S6_L003.sam
        (one of the commands exited with non-zero exit code; note that snakemake uses bash strict mode!)
    cluster_jobid: Submitted batch job 19120477

Error executing rule BWA on cluster (jobid: 25, external: Submitted batch job 19120477, jobscript: /cluster/scratch/michalo/Luca_dna/.snakemake/tmp.3_v3ozd
f/snakejob.BWA.25.sh). For error details see the cluster log and the log files of the involved rule(s).
[Wed Jun 21 12:11:15 2023]
Finished job 11.
3 of 62 steps (5%) done
[Wed Jun 21 12:36:32 2023]
Finished job 46.
4 of 62 steps (6%) done
[Wed Jun 21 12:48:46 2023]
Finished job 61.
5 of 62 steps (8%) done
```

Workflow execution – a happy finish

```
module load samtools
samtools index sorted_reads/Beta5-genomicDNA_S4_L003.bam
Submitted job 27 with external jobid 'Submitted batch job 19219893'.
[Thu Jun 22 16:38:40 2023]
Finished job 27.
24 of 26 steps (92%) done
Select jobs to execute...

[Thu Jun 22 16:38:40 2023]
localrule summary_bai:
    input: sorted_reads/alphaV-genomicDNA_S1_L003.bam.bai, sorted_reads/AlphaVBeta1-genomicDNA_S5_L003.bam.bai, sorted_reads/WildTypeFibroblastWTF-
genomicDNA_S6_L003.bam.bai, sorted_reads/Beta3-genomicDNA_S3_L003.bam.bai, sorted_reads/Beta1-genomicDNA_S2_L003.bam.bai, sorted_reads/Beta5-genomi
cDNA_S4_L003.bam.bai
    output: secondary_analysis/final_marker_bai.txt
    jobid: 1
    reason: Missing output files: secondary_analysis/final_marker_bai.txt; Input files updated by another job: sorted_reads/AlphaVBeta1-genomicDNA_
S5_L003.bam.bai, sorted_reads/Beta5-genomicDNA_S4_L003.bam.bai, sorted_reads/alphaV-genomicDNA_S1_L003.bam.bai, sorted_reads/WildTypeFibroblastWTF-
genomicDNA_S6_L003.bam.bai, sorted_reads/Beta1-genomicDNA_S2_L003.bam.bai, sorted_reads/Beta3-genomicDNA_S3_L003.bam.bai
    resources: mem_mb=1000, mem_mib=954, disk_mb=1000, disk_mib=954, tmpdir=/tmp

touch secondary_analysis/final_marker_bai.txt
[Thu Jun 22 16:38:40 2023]
Finished job 1.
25 of 26 steps (96%) done
Select jobs to execute...

[Thu Jun 22 16:38:40 2023]
localrule all:
    input: secondary_analysis/final_marker_bai.txt
    jobid: 0
    reason: Input files updated by another job: secondary_analysis/final_marker_bai.txt
    resources: mem_mb=1000, mem_mib=954, disk_mb=1000, disk_mib=954, tmpdir=/tmp

[Thu Jun 22 16:38:40 2023]
Finished job 0.
26 of 26 steps (100%) done
Complete log: .snakemake/log/2023-06-22T134327.064185.snakemake.log
(snakemake) [michalo@eu-login-05 Luca_dna]$
```

Workflow development, optimization and scalability

Workflow development, optimization and scalability

- Most of the stories from this presentation of Sam Fux are still valid !

Workflow development hints

- Develop directly in the final environment, eg on the cluster
  - switching the software stack may be confusing
  - the same storage location and type

- Reuse templates as much as possible
  - for the whole workflow and for the individual rules
  - syntax errors happen to be tricky to trace. Typical: a missing colon.
  - Tricky system of escape characters, eg

```
shell("""  awk '/class_code \"u\"/{{print}}' stringtie/{wildcards.sample}.annotated.gtf >
stringtie/{wildcards.sample}.novel.gtf  \n """)
```

Workflow optimization hints

- In case of time/memory optimization needs for a rule, try it out of the workflow in a single job.
- Use the Slurm Submission Line Advisor
- Use the optimal storage type for your data

File system	Life span	Backup	Max size	Small files	Large files
`/cluster/home`	Permanent	Yes	16 GB	+	o
`/cluster/scratch`	2 weeks	No	2.5 TB	o	++
`/cluster/project`	4 years	Optional	Flexible	+	+
`/cluster/work`	4 years	No	Flexible	o	++
local `/scratch`	Job	No	800 GB	++	o
central NAS	Flexible	Optional	Flexible	+	+

Workflow debugging hints

- In case of a crash of a rule, try it out of the workflow in a single job.
  - Use the command line from the Slurm log
- Typical reasons for crashing:
  - Syntax errors in the executable part of the rule
  - Wrong input/output definitions in the rule
  - Wrongly configured memory, cores or running timetime for a job
- Fix the rule, and re-try the whole workflow using the existing intermediate files
- Use python-style prints for debugging in the freeform python and rules

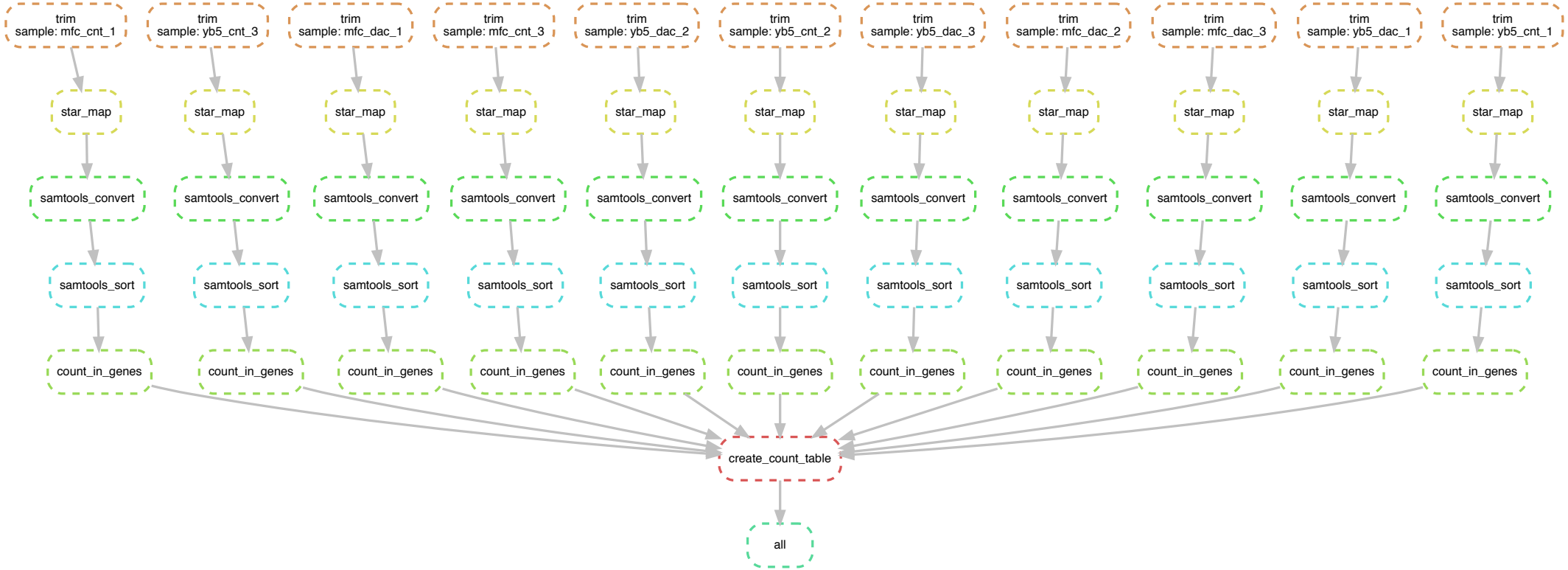
Workflow scalability hints

- For a start, prepare a small set of test data
  - Eg. 3 input files
  - Prepare the subsets for the big data, eg in sequencing files just first 10k-1M sequencer reads
- Do not overuse multiple wildcards
  - Increases computational complexity, decreases readability
  - Check the graph in case of doubt
  - Safe and manageable: up to 2 wildcards
- Number of items in the wildcard is typically not a problem
  - Snakemake can be a more controllable replacement for job arrays by “naive map-reduce”

Workflow case studies

Typical RNA-seq primary analysis

- Embarrasingly parallel: from sequencing samples to the count table



Current template: https://github.com/michalogit/snake_hisat

Virsorter2 – Snakemake-based metabolomics combo software

BMC Part of Springer Nature

Microbiome

Home About Articles Sections Collections Submission Guidelines Submit manuscript

Software article | Open Access | Published: 01 February 2021

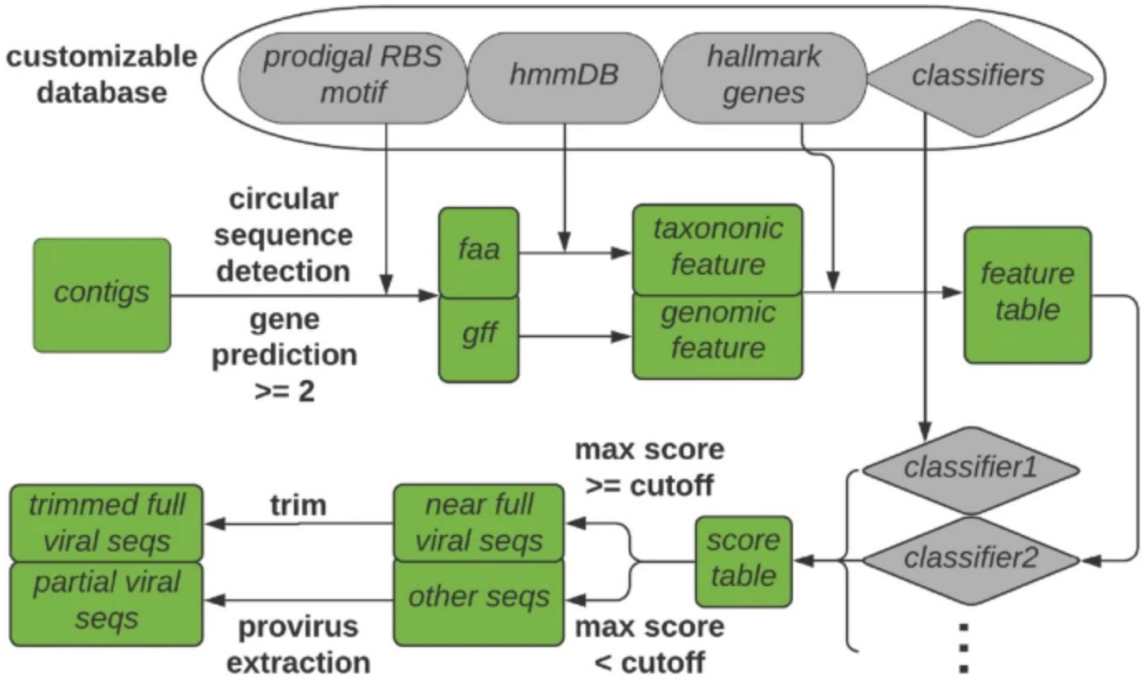
VirSorter2: a multi-classifier, expert-guided approach to detect diverse DNA and RNA viruses

Jiarong Guo, Ben Bolduc, Ahmed A. Zayed, Arvind Varsani, Guillermo Dominguez-Huerta, Tom O. Delmont, Akbar Adjie Pratama, M. Consuelo Gazitúa, Dean Vik, Matthew B. Sullivan & Simon Roux

Microbiome **9**, Article number: 37 (2021) | Cite this article

21k Accesses | **210** Citations | **77** Altmetric | Metrics

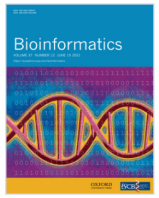
Fig. 1



V-pipe – pipeline for COVID samples processing

Bioinformatics

Issues Advance articles Submit ▾ Alerts About ▾ Bioinformatics



Volume 37, Issue 12
June 2021

Article Contents

JOURNAL ARTICLE

V-pipe: a computational pipeline for assessing viral genetic diversity from high-throughput data

Susana Posada-Céspedes, David Seifert, Ivan Topolsky, Kim Philipp Jablonski, Karin J Metzner, Niko Beerenwinkel ✉

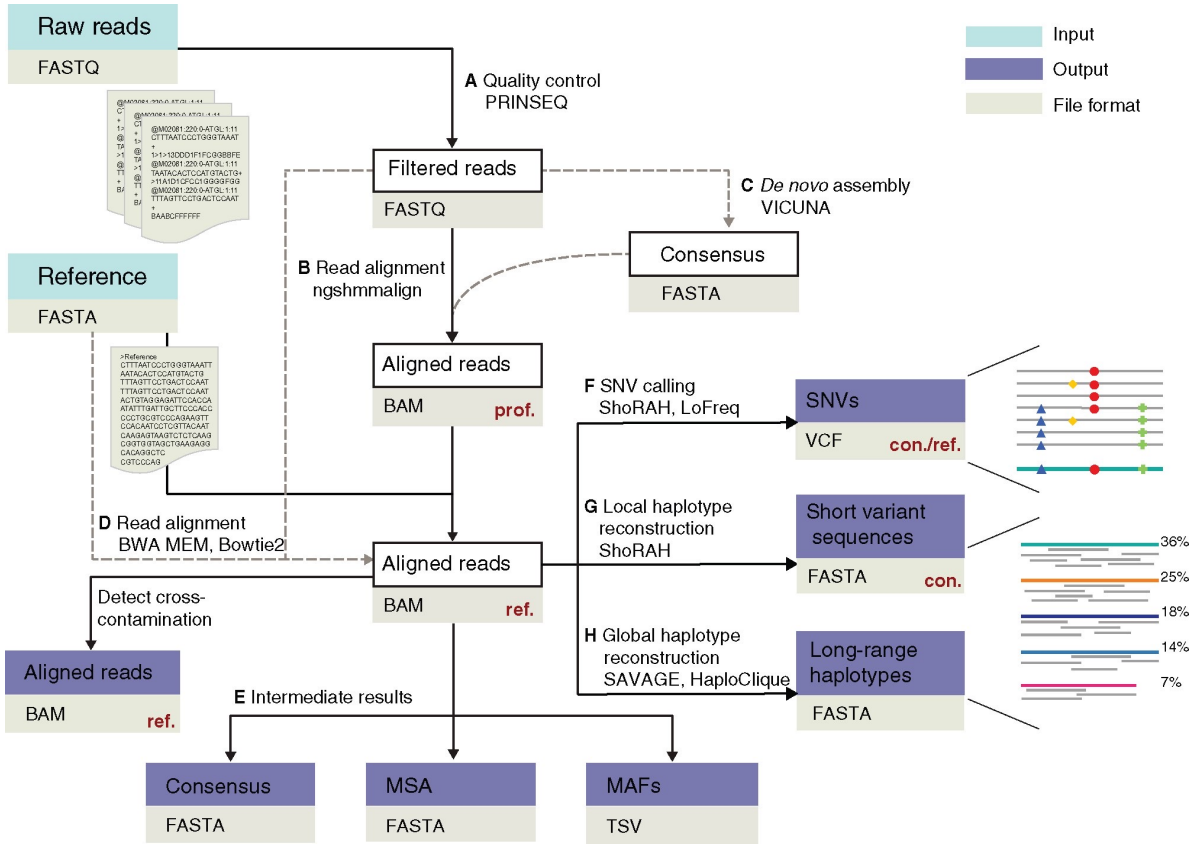
Bioinformatics, Volume 37, Issue 12, June 2021, Pages 1673–1680,
https://doi.org/10.1093/bioinformatics/btab015
Published: 20 January 2021 **Article history** ▾

PDF Split View Cite Permissions Share ▾

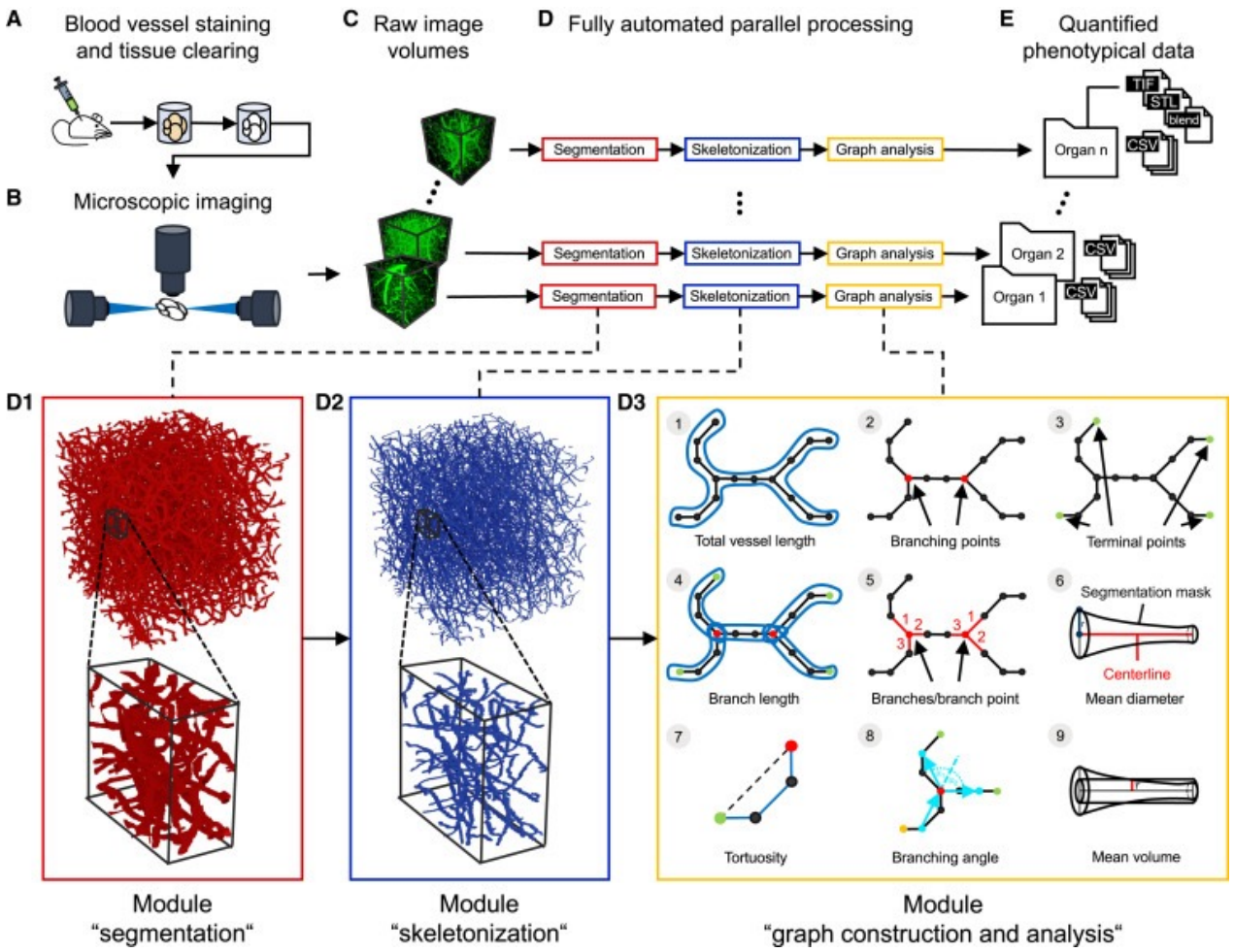
Abstract

Motivation

High-throughput sequencing technologies are used increasingly not only in viral genomics research but also in clinical surveillance and diagnostics. These technologies facilitate the assessment of the genetic diversity in intra-host virus populations, which affects transmission, virulence and pathogenesis of



Applications in image analysis



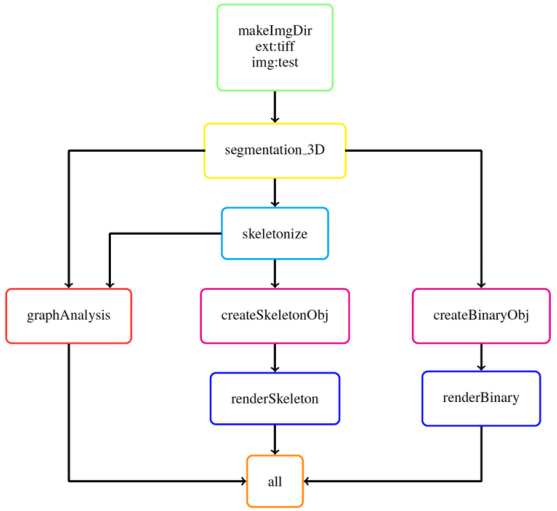
Cell Reports Methods

Volume 3, Issue 3, 27 March 2023, 100436

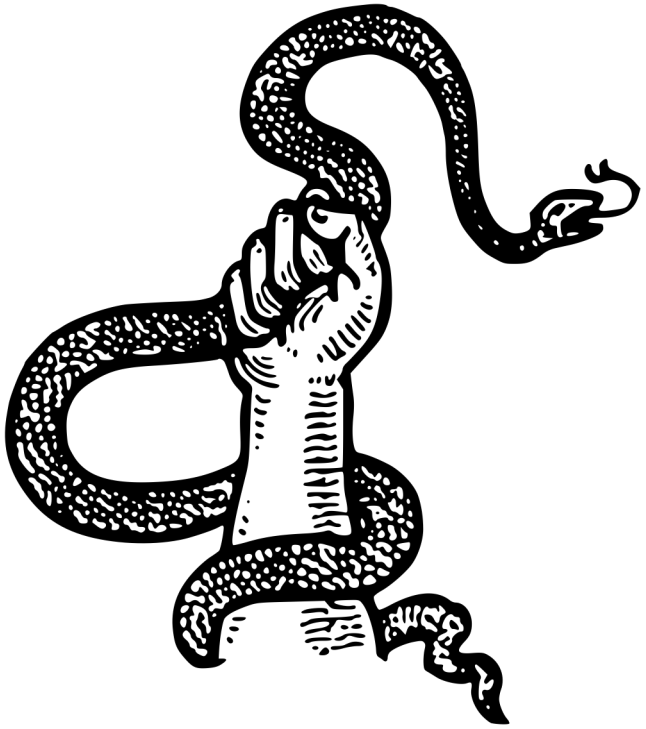
Article

Rapid and fully automated blood vasculature analysis in 3D light-sheet image volumes of different organs

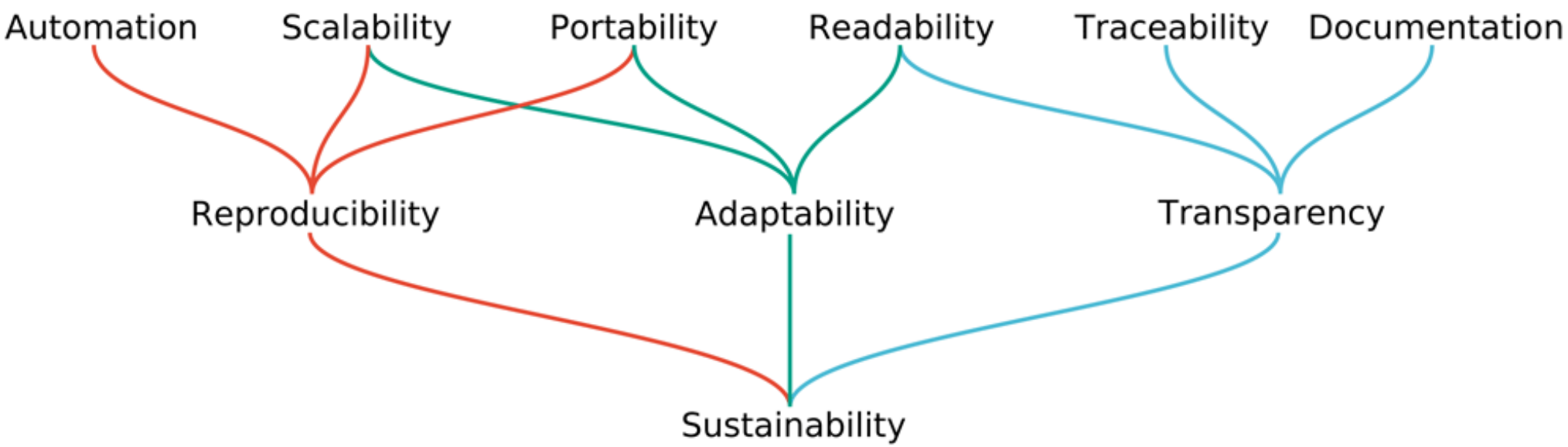
Philippa Spangenberg [1 4 10], Nina Hagemann [2 10], Anthony Squire [3], Nils Förster [4 9], Sascha D. Krauß [3], Yachao Qi [2], Ayan Mohamud Yusuf [2], Jing Wang [2], Anika Grüneboom [5], Lennart Kowitz [5], Sebastian Korste [6], Matthias Totzeck [6], Zülal Cibir [3], Ali Ata Tuz [3], Vikramjeet Singh [3], Devon Siemes [1], Laura Struensee [4], Daniel R. Engel [1], Peter Ludewig [7], Luiza Martins Nascentes Melo [8]...Axel Mosig [4 9 10 11]



Summary



Snakemake as a technique for “sustainable data analysis”



F1000Res. 2021 Jan 18;10:33. doi: 10.12688/f1000research.29032.2. eCollection 2021.

Sustainable data analysis with Snakemake

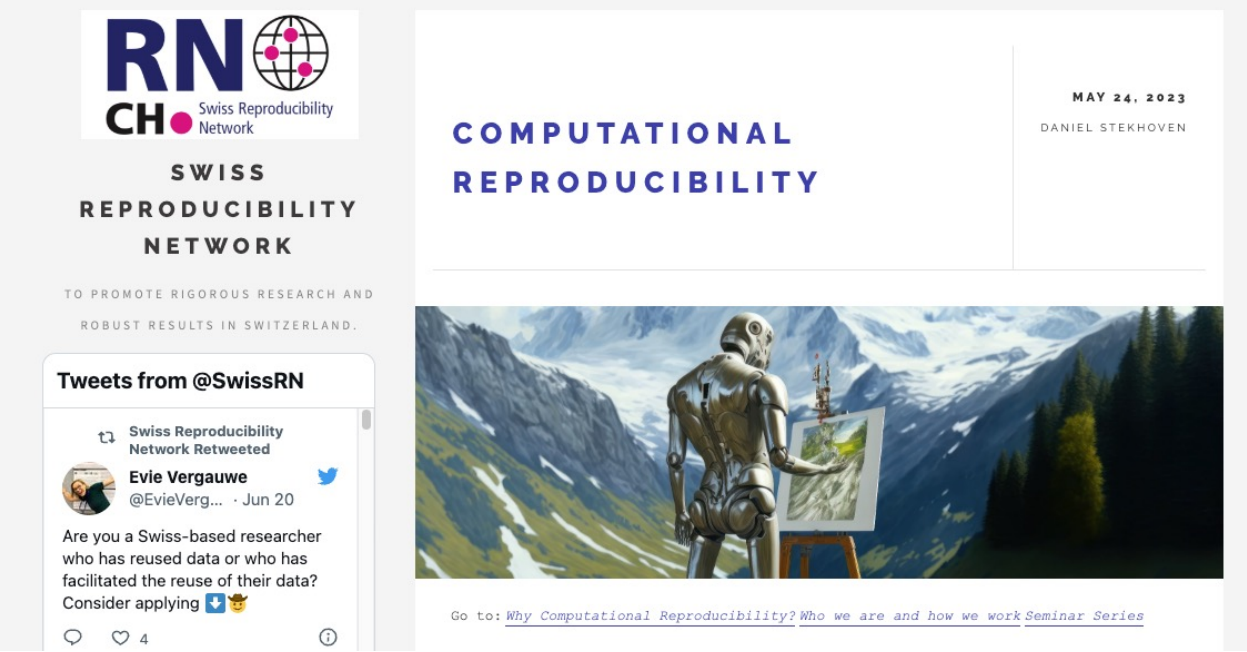
Felix Mölder [1 2], Kim Philipp Jablonski [3 4], Brice Letcher [5], Michael B Hall [5], Christopher H Tomkins-Tinch [6 7], Vanessa Sochat [8], Jan Forster [1 9], Soohyun Lee [10], Sven O Twardziok [11], Alexander Kanitz [12 13], Andreas Wilm [14], Manuel Holtgrewe [11 15], Sven Rahmann [16], Sven Nahnsen [17], Johannes Köster [1 18]

Affiliations + expand

PMID: 34035898 PMCID: PMC8114187 DOI: 10.12688/f1000research.29032.2

Free PMC article

Reproducibility seminar by Johannes Köster…



Date	Topic	Speaker
2023-10-18	Multi-tool deployment using containers	Vipin Sreedharan, ETH Zurich
2023-11-15	MaRDI TA2: Research Data and Reproducibility in Scientific Computing	Jens Saak, MPI for Dynamics of Complex Technical Systems
2023-12-13	Sustainable data science with the Renku platform	Rok Roškar, Swiss Data Science Center
2024-01-17	The Reproducible Research Platform – Towards FAIR and reproducible sharing of data, code and computational environments	Henry Lütcke, ETH Zurich
2024-02-21	TBA	TBA
2024-03-20	TBA	TBA
TBA	Reproducibility and beyond with Snakemake and Datavzrd	Johannes Köster, Uni Duisburg Essen

Why snakemake?

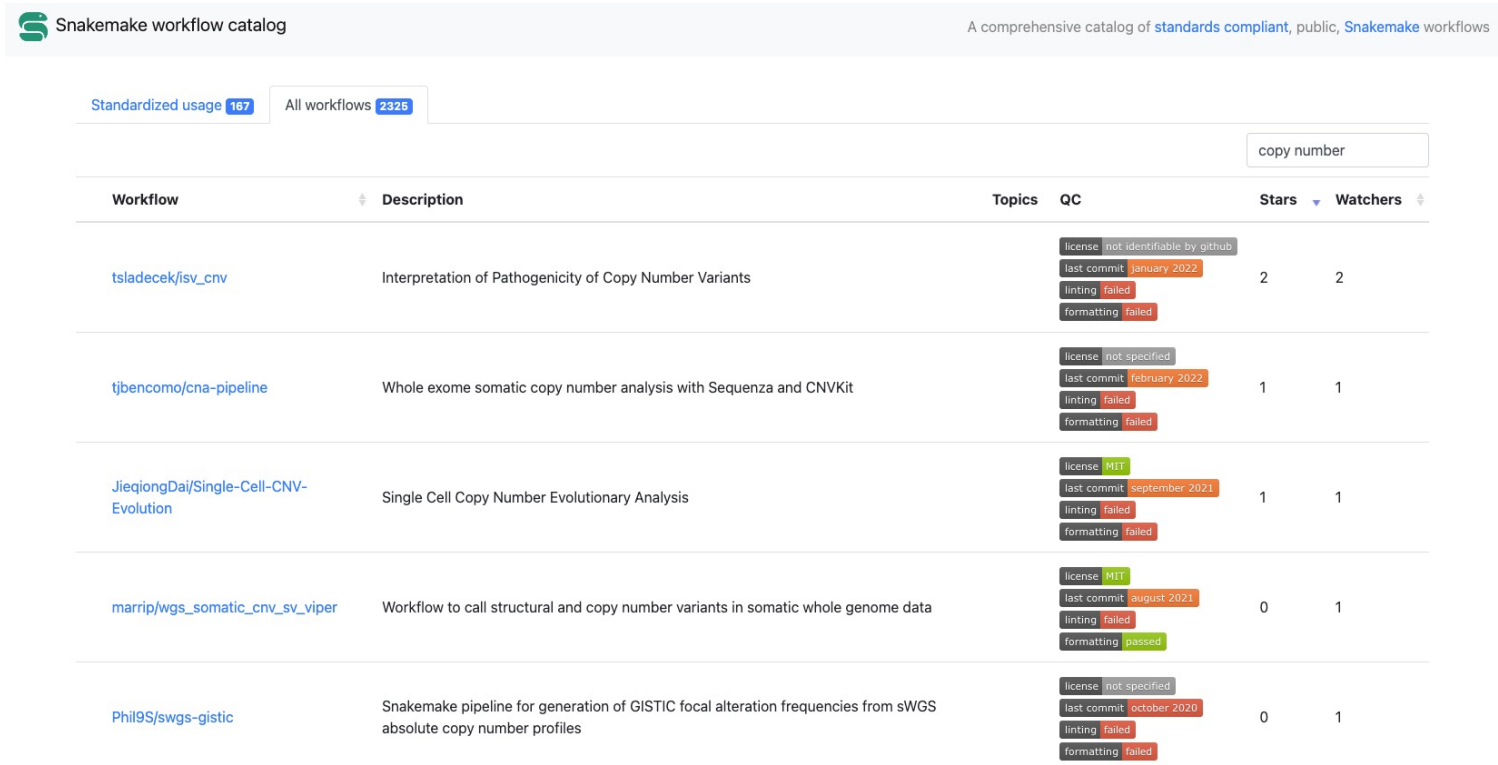
- It provides a good balance between reproducibility and flexibility
  - One can share a workflow and ask to reproduce it
  - Workflows are publishable
  - Workflow templates are relatively easy to experiment with and adapt
- Workflows are human-readable as sets of rules and graphs
  - Users (biologists) can verify what was done in the processing

Why snakemake?

- Snakemake can re-start after a crash
  - Re-use the partial product
  - Pick up after a fixing, where it has stopped
  - It needs to be done carefully
- Snakemake can solve a number of processing tasks of various complexity
  - Adaptation of templates is efficient, having some experience
- If you can think of a feature in a workflow, Johannes Köster and his team had likely analyzed, tested and implemented it before

Why snakemake?

- Community and the amount of workflow resources is growing
  - https://snakemake.github.io/snakemake-workflow-catalog/



ETH zürich

Thank you for your attention!

Snakemake introduction