Swiss Institute of
Bioinformatics

Nextflow in Action:

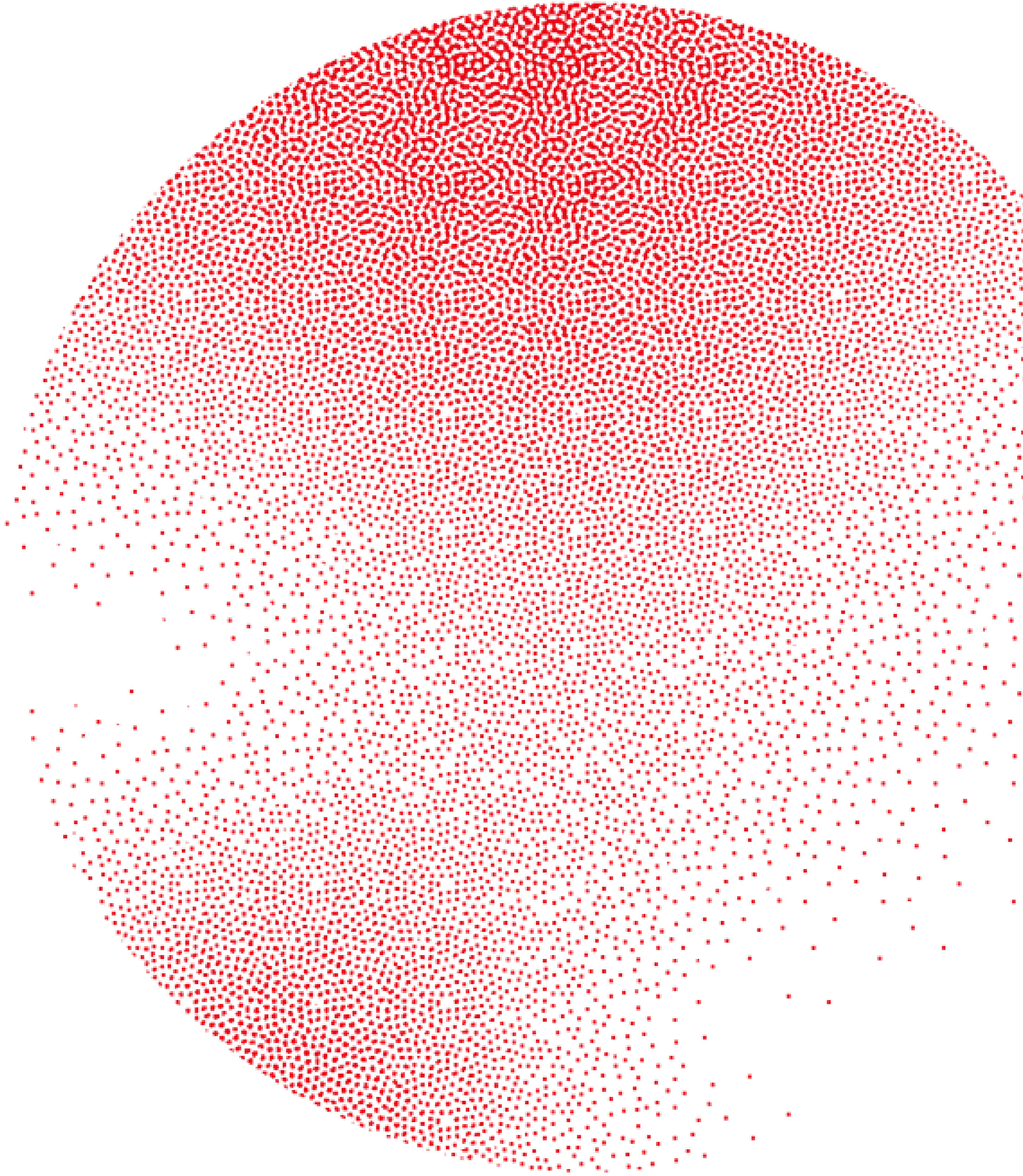
Build Smarter, Faster, Reproducible Pipelines

PhDc Jeferyd Yepes-García
University of Fribourg
BUGFri
March 17th, 2026

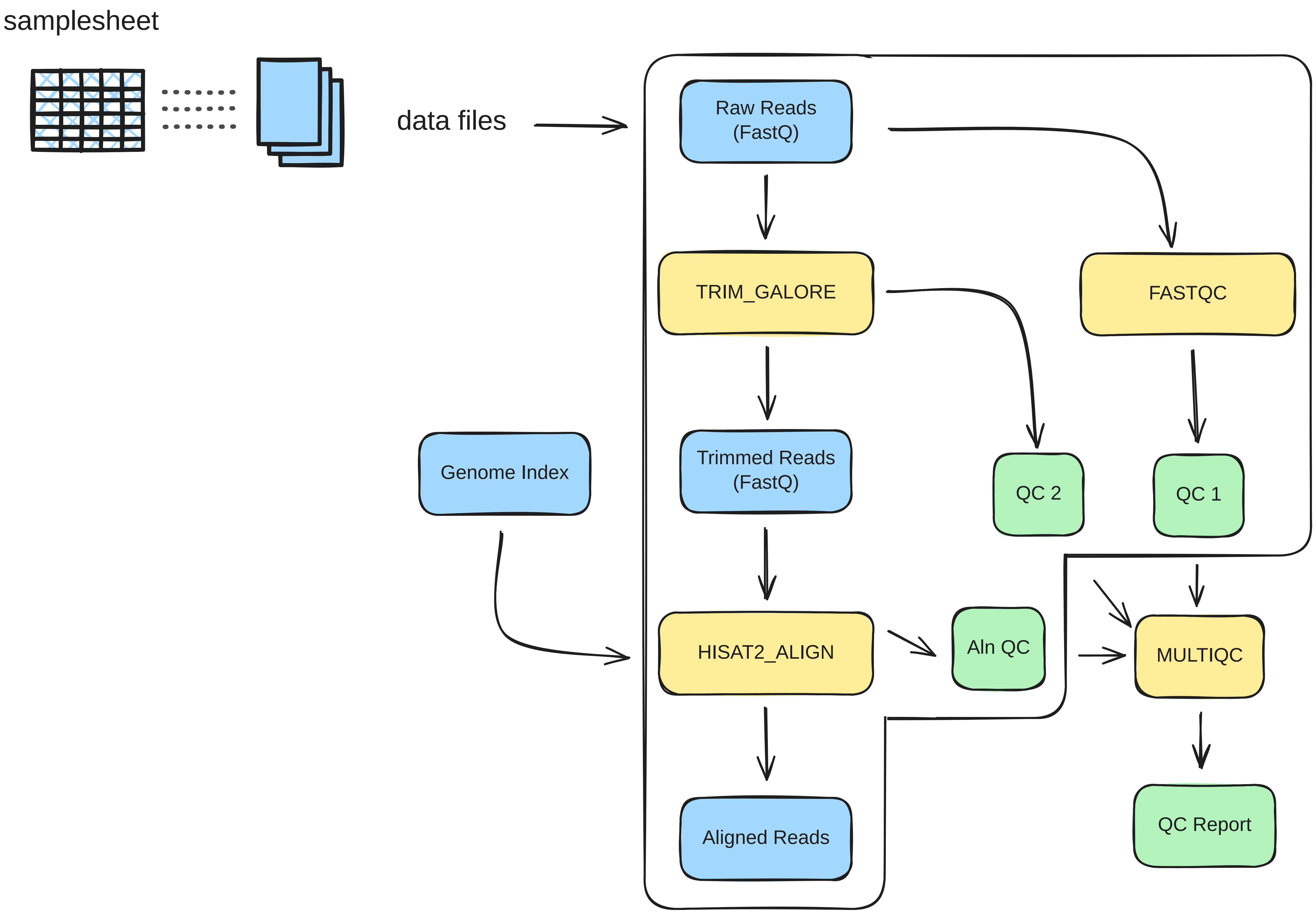
elixir SWITZERLAND

This presentation

- RNA-seq pipeline
- Operators
- Executors
- Software management
- Profiles



The pipeline



samplesheet

data files

Raw Reads (FastQ)

TRIM_GALORE

FASTQC

Genome Index

Trimmed Reads (FastQ)

QC 2

QC 1

HISAT2_ALIGN

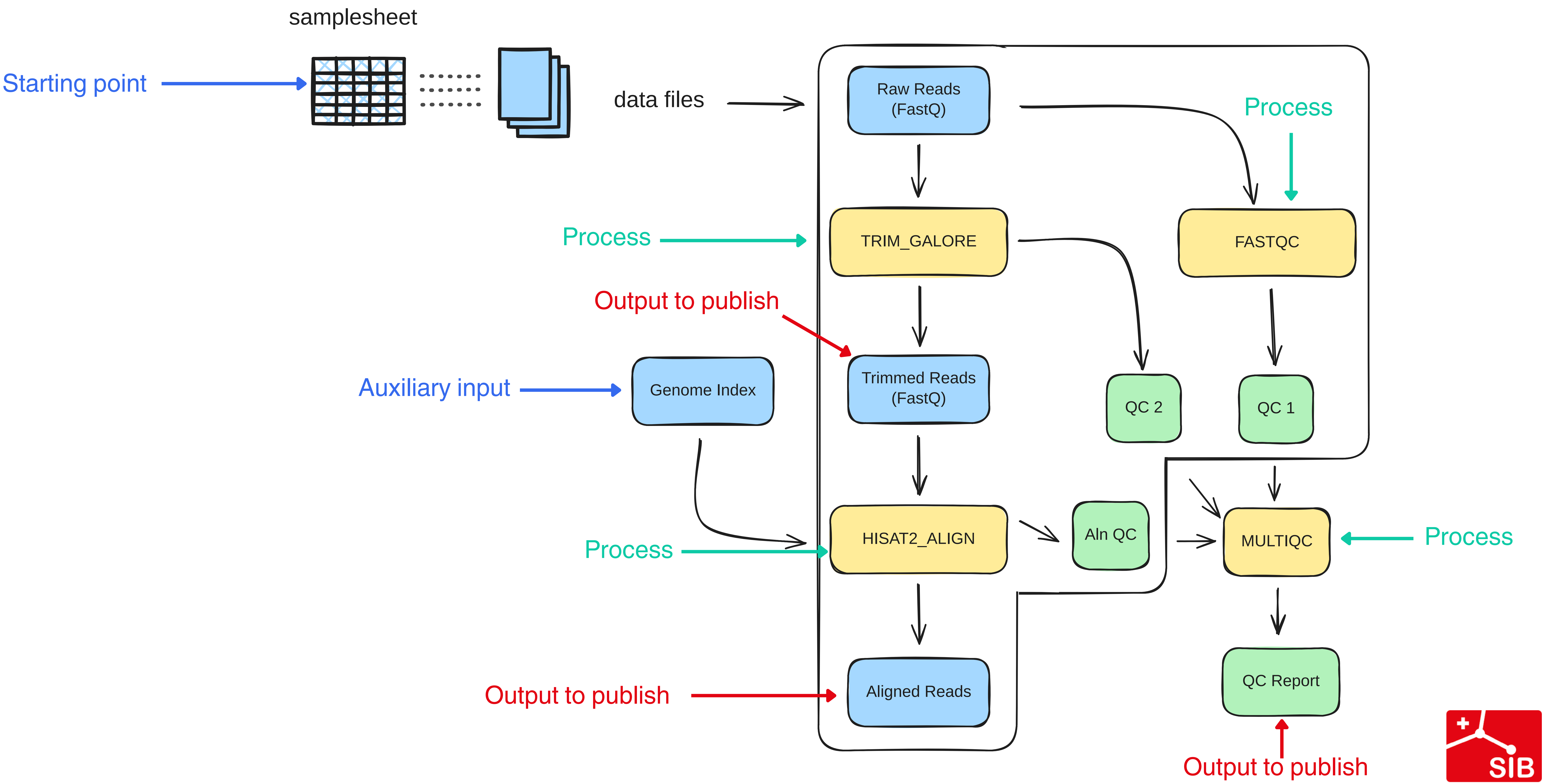
Aln QC

MULTIQC

Aligned Reads

QC Report

The pipeline



samplesheet

Starting point

data files

Raw Reads
(FastQ)

Process

Process

TRIM_GALORE

FASTQC

Output to publish

Auxiliary input

Genome Index

Trimmed Reads
(FastQ)

QC 2

QC 1

Process

HISAT2_ALIGN

Aln QC

MULTIQC

Process

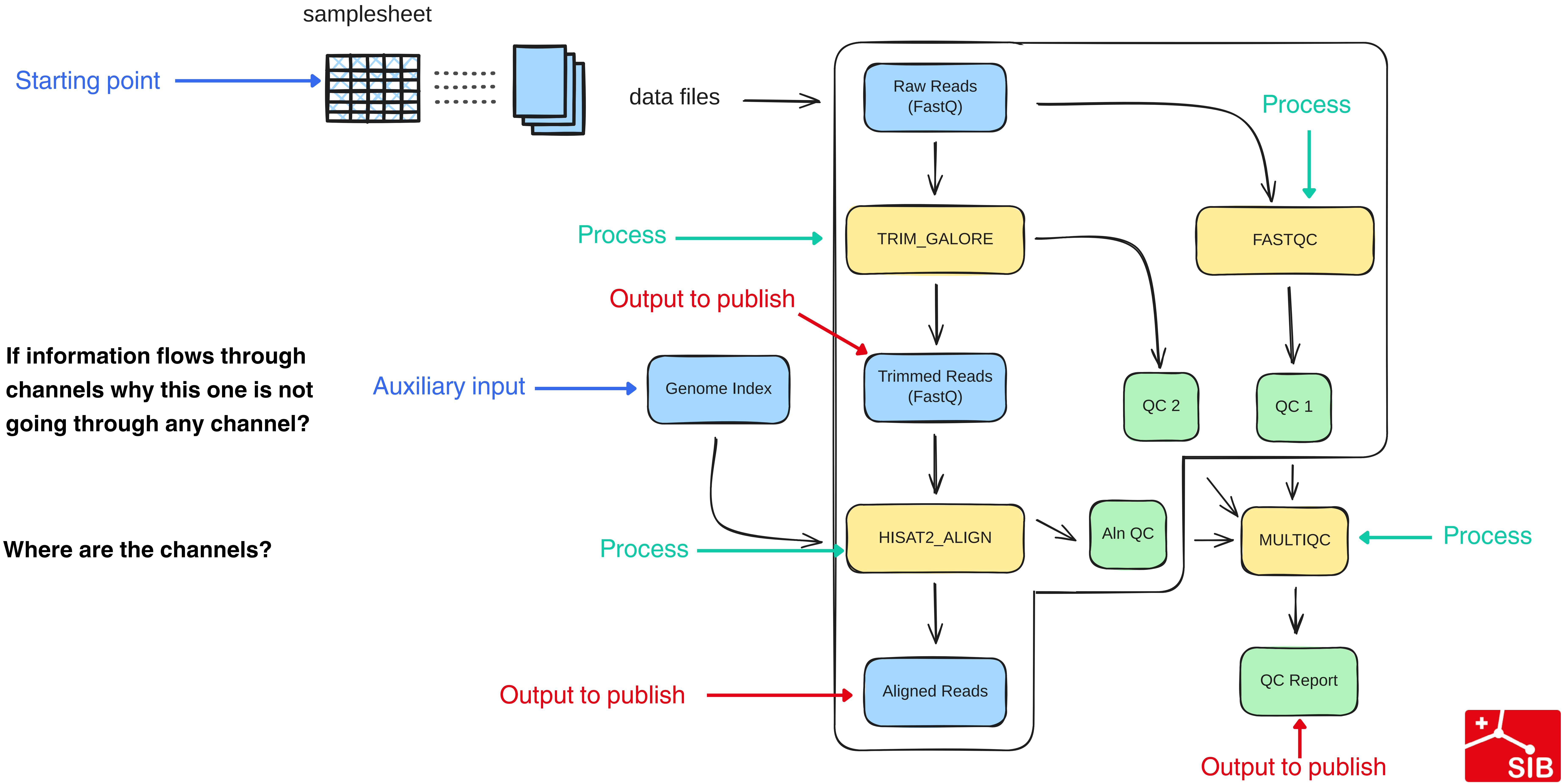
Output to publish

Aligned Reads

QC Report

Output to publish

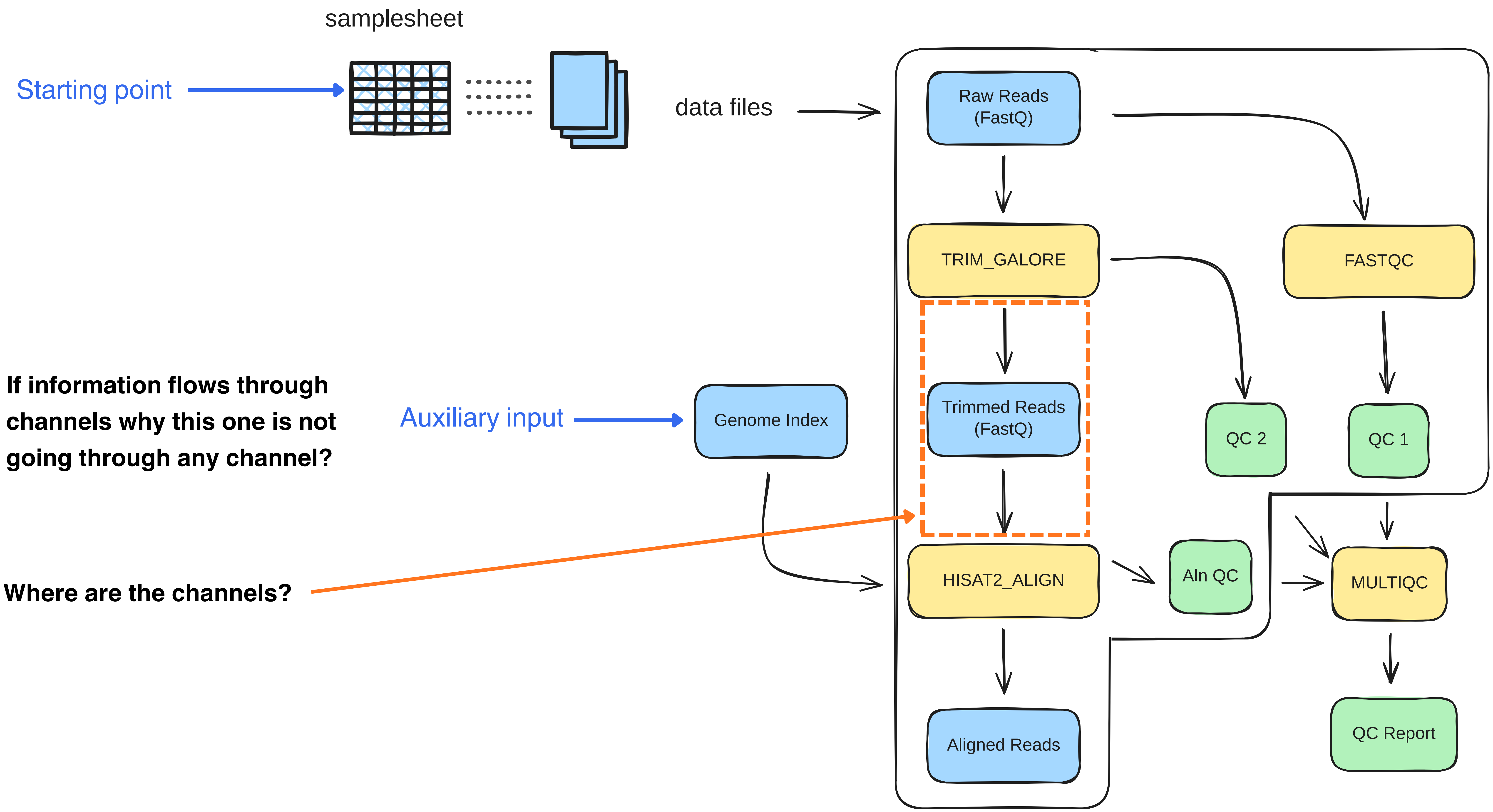
The pipeline



If information flows through channels why this one is not going through any channel?

Where are the channels?

The pipeline



If information flows through channels why this one is not going through any channel?

Where are the channels?

Operators: shaping and organizing data

Most common operators:

- view()

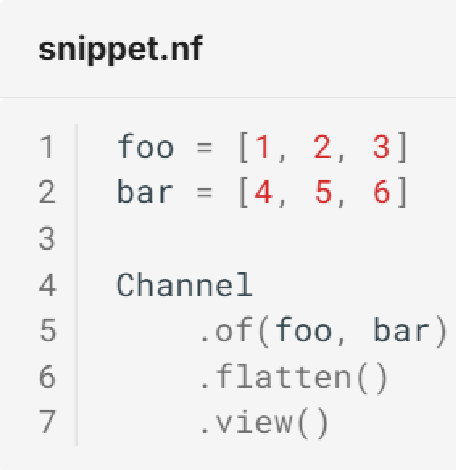
snippet.nf

```
Channel
    .of('foo', 'bar', 'baz')
    .view()
```

Output

```
foo
bar
baz
```

- flatten()



snippet.nf

```
foo = [1, 2, 3]
bar = [4, 5, 6]

Channel
    .of(foo, bar)
    .flatten()
    .view()
```

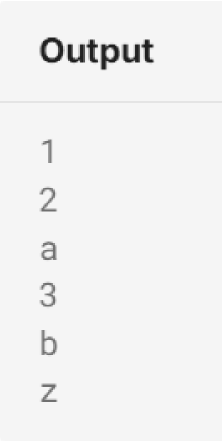
Output

```
1
2
3
4
5
6
```

- mix()

snippet.nf

```
my_channel_1 = Channel.of(1, 2, 3)
my_channel_2 = Channel.of('a', 'b')
my_channel_3 = Channel.of('z')

my_channel_1
    .mix(my_channel_2, my_channel_3)
    .view()
```



Output

```
1
2
a
3
b
z
```

- collect()

snippet.nf

```
Channel
    .of(1, 2, 3, 4)
    .collect()
    .view()
```

Output

```
[1, 2, 3, 4]
```

Operators: shaping and organizing data

Most common operators:

- view()

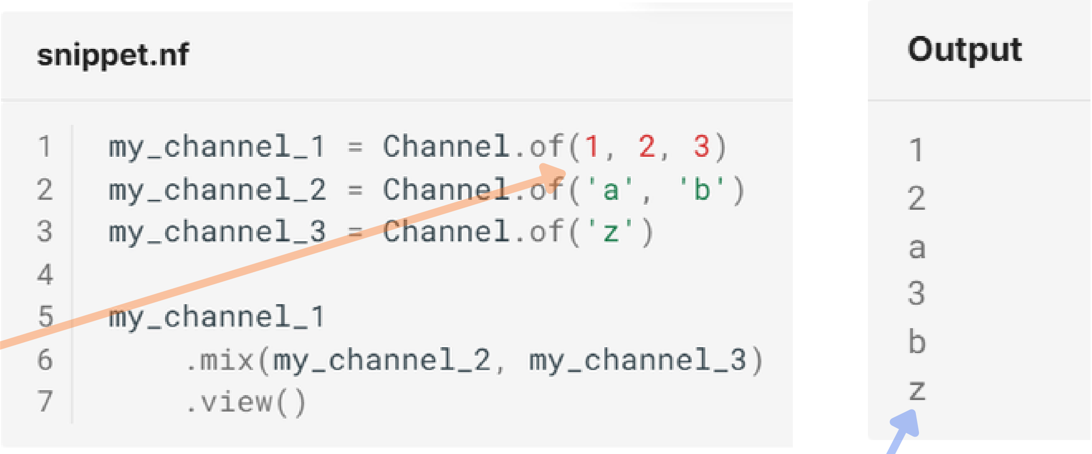
snippet.nf

```
Channel
    .of('foo', 'bar', 'baz')
    .view()
```

Output

```
foo
bar
baz
```

- mix()



snippet.nf

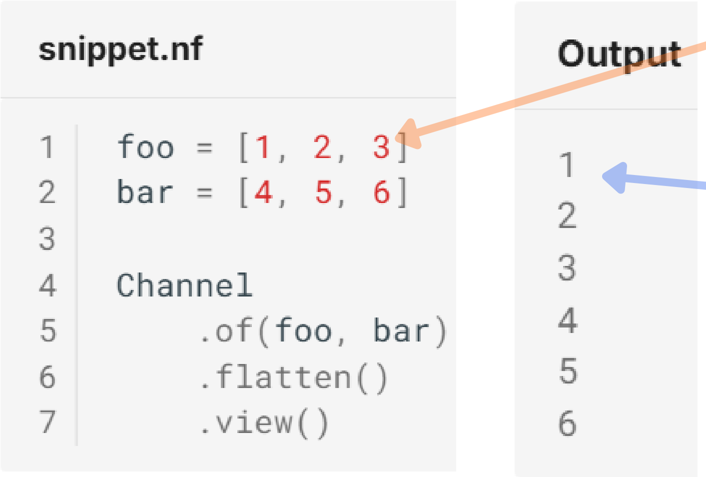
```
my_channel_1 = Channel.of(1, 2, 3)
my_channel_2 = Channel.of('a', 'b')
my_channel_3 = Channel.of('z')

my_channel_1
    .mix(my_channel_2, my_channel_3)
    .view()
```

Output

```
1
2
a
3
b
z
```

The shape of the **input** is different

- flatten()



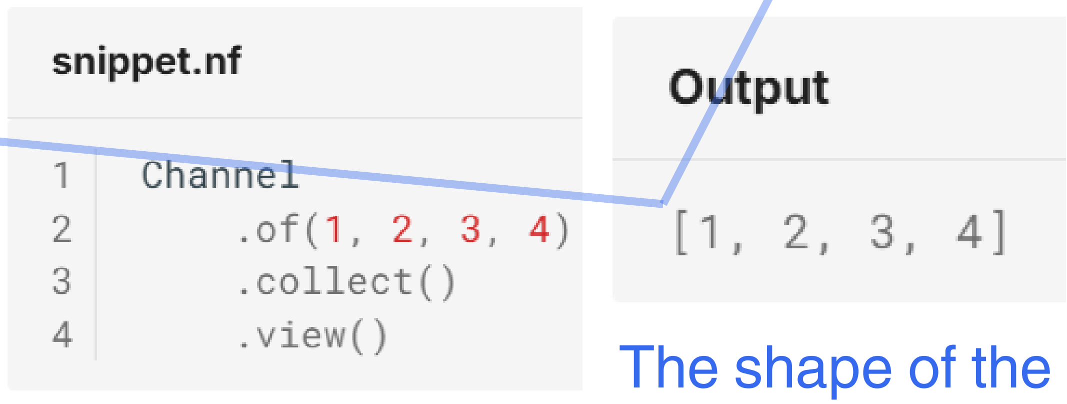
snippet.nf

```
foo = [1, 2, 3]
bar = [4, 5, 6]

Channel
    .of(foo, bar)
    .flatten()
    .view()
```

Output

```
1
2
3
4
5
6
```

- collect()



snippet.nf

```
Channel
    .of(1, 2, 3, 4)
    .collect()
    .view()
```

Output

```
[1, 2, 3, 4]
```

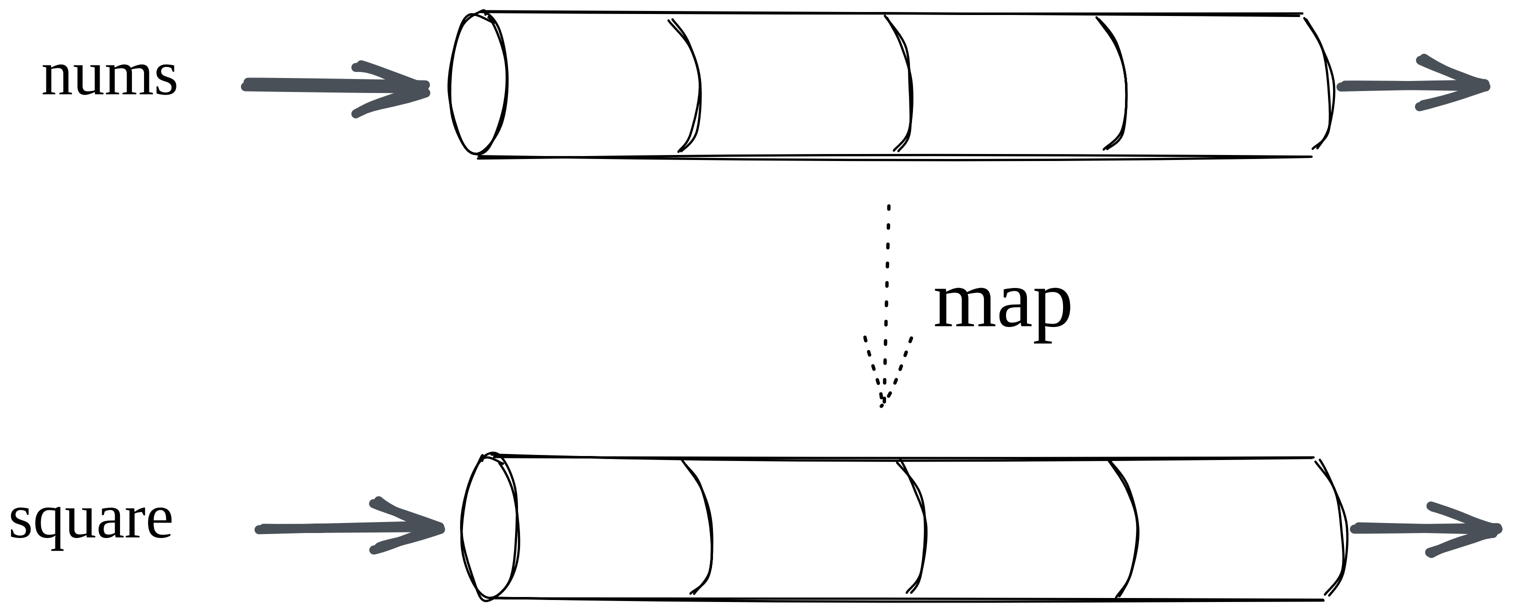
The shape of the **output** is different

Operators: shaping and organizing data

- map()

```
nums = channel.of(1, 2, 3, 4)
square = nums.map { num -> num * num }
square.view()
```

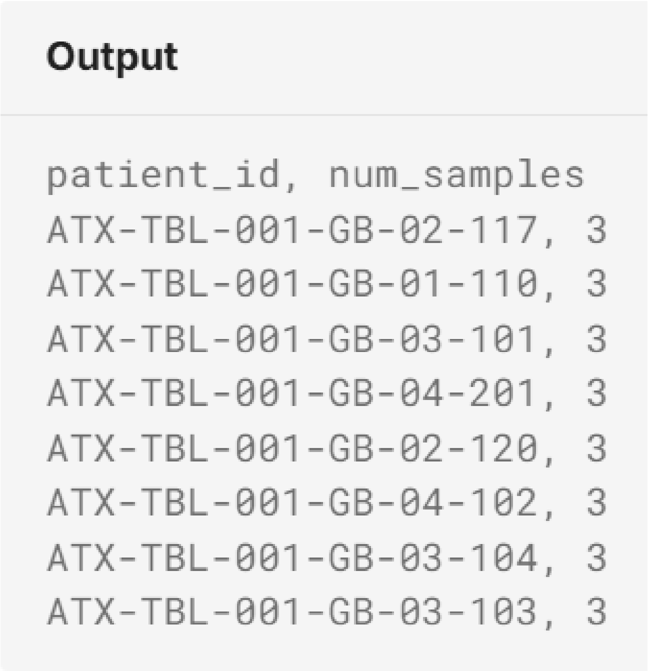
Closure



- splitCsv()

snippet.nf

```
Channel
    .fromPath("data/meta/patients_1.csv")
    .splitCsv()
    .view { row -> "${row[0]}, ${row[3]}" }
```



Output

```
patient_id, num_samples
ATX-TBL-001-GB-02-117, 3
ATX-TBL-001-GB-01-110, 3
ATX-TBL-001-GB-03-101, 3
ATX-TBL-001-GB-04-201, 3
ATX-TBL-001-GB-02-120, 3
ATX-TBL-001-GB-04-102, 3
ATX-TBL-001-GB-03-104, 3
ATX-TBL-001-GB-03-103, 3
```

Executors

- How the execution is controlled, resources to allocate and task orchestration.

`nextflow.config`

```
process.executor = ‘local’
process.executor = ‘slurm’
process.executor = ‘sge’
process.executor = ‘aws batch’
...
```

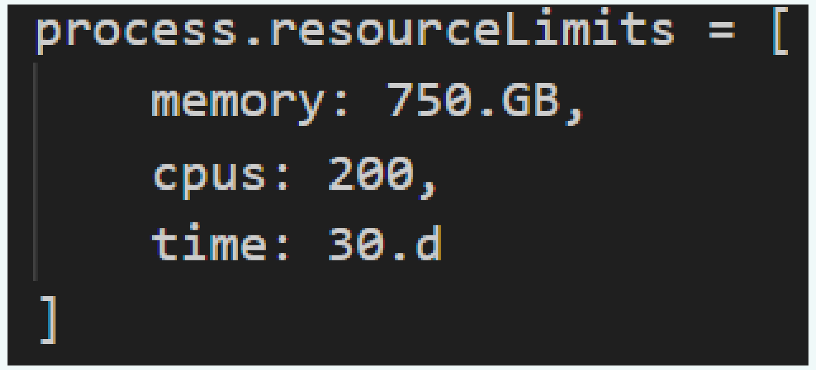
Executors

- How the execution is controlled, resources to allocate and task orchestration.

nextflow.config

```
process.executor = ‘local’
process.executor = ‘slurm’
process.executor = ‘sge’
process.executor = ‘aws batch’
...
```

Resource limits



```
process.resourceLimits = [
    memory: 750.GB,
    cpus: 200,
    time: 30.d
]
```

Software management

- What software is required, where it is going to be retrieved from and execution environment.

Mamba

CONDA

docker

podman

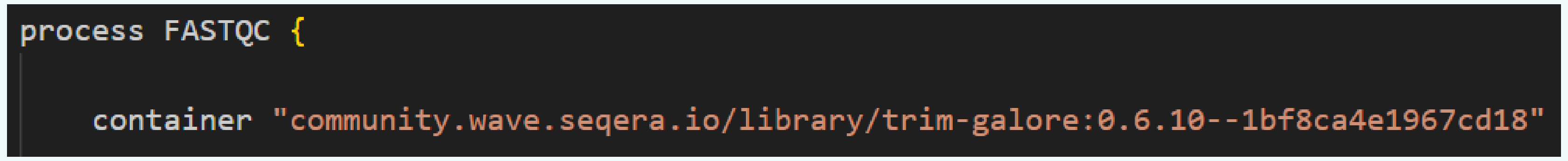
Charliecloud

Spack Packages

nextflow.config

```
docker.enabled = true
conda.enabled = true
apptainer.enabled = true
...
```

Container



```
process FASTQC {

    container "community.wave.seqera.io/library/trim-galore:0.6.10--1bf8ca4e1967cd18"
```

Conda

```
conda 'bioconda::fastqc=0.12.1'
```

Software management

- Important considerations:

Mamba

CONDA

docker

podman

Charliecloud

Spack Packages

Docker

Nextflow **automatically manages the file system mounts** whenever a container is launched depending on the process input files.

Conda

Nextflow does not report when a Conda environment creation fails.

Apptainer

Unlike Docker, Nextflow **does not automatically mount host paths** in the container when using Apptainer. It expects that the paths are configured and mounted system wide by the Apptainer runtime.

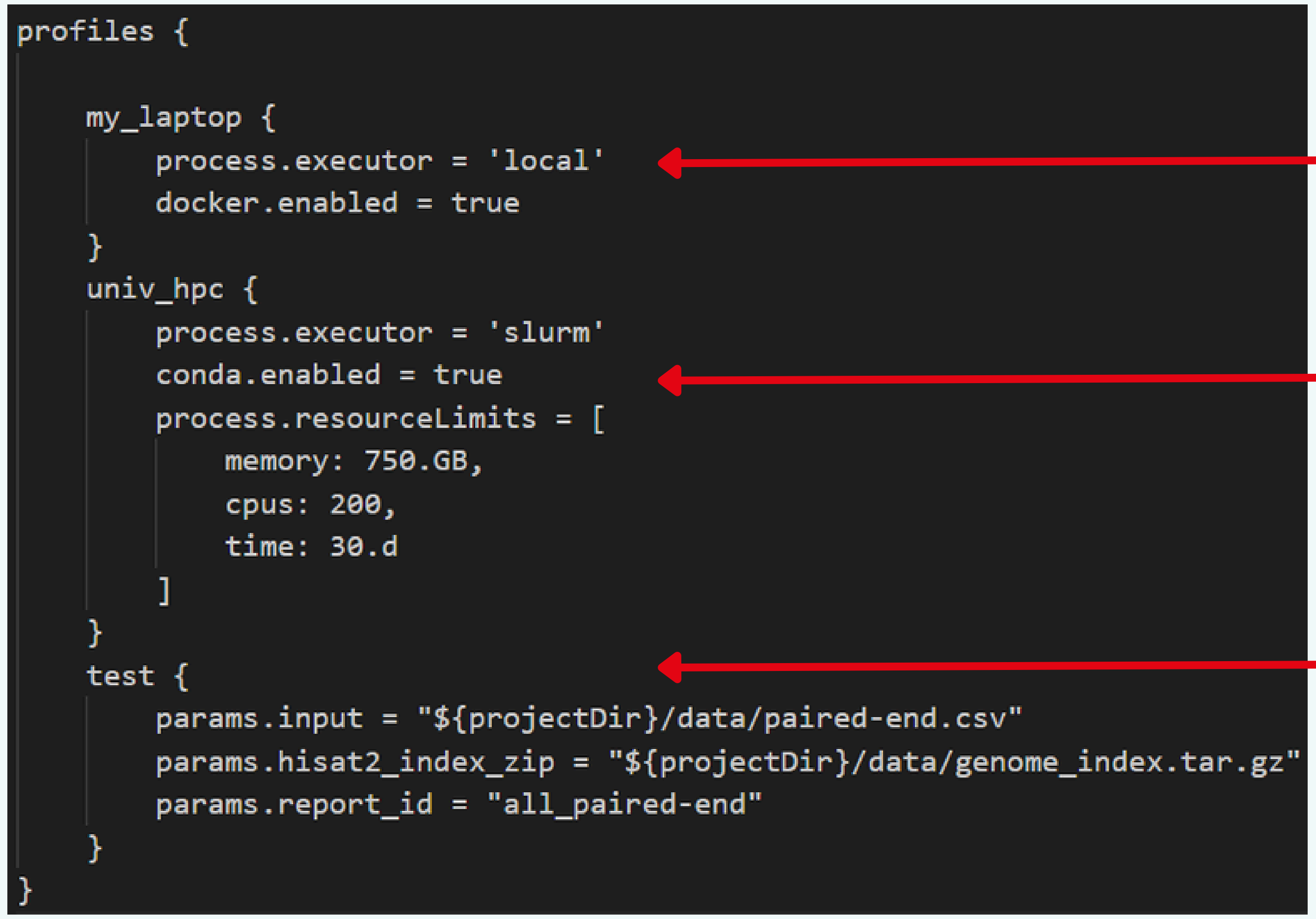
Although possible.

Do not blend technologies!

Profiles: combining everything

- Profiles gather parameters, and software execution and managenement.

nextflow.config



```
profiles {

    my_laptop {
        process.executor = 'local'
        docker.enabled = true
    }
    univ_hpc {
        process.executor = 'slurm'
        conda.enabled = true
        process.resourceLimits = [
            memory: 750.GB,
            cpus: 200,
            time: 30.d
        ]
    }
    test {
        params.input = "${projectDir}/data/paired-end.csv"
        params.hisat2_index_zip = "${projectDir}/data/genome_index.tar.gz"
        params.report_id = "all_paired-end"
    }
}
```

Profile aiming at local runtime

Profile aiming at HPC execution

Profile for testing and debugging

CLI

Profiles can be mixed.
Make sure you don’t override directives.

It's your time!