Making reproducible workflows with

# nextflow

South Green bioinformatics platform

## Nextflow features

- Generalisable
- Portable
- Scalable
- Platform-agnostic
- Based on Groovy and Java
- Large active community in e.g. nf-core

## Concepts and nomenclature

- Channels contain data, e.g. input files
- Processes run some kind of code, e.g. a script or a command-line program
- Tasks are instances of a process, one per process input

## Anatomy of a process

```
process GET_SRA_BY_ACCESSION {

    input:
    val(sample)

    output:
    path("${sample}.fastq.gz")

    script:
    """
    fastq-dump ${sample} > ${sample}.fastq.gz
    """
}
```

## Anatomy of a process

```
process GET_SRA_BY_ACCESSION {

    input:
    val(sample)

    output:
    tuple val(sample), path("${sample}.fastq.gz")

    script:
    """
    fastq-dump ${sample} > ${sample}.fastq.gz
    """
}
```

# Anatomy of a process

```
process GET_SRA_BY_ACCESSION {

    cpus 2
    memory '8 GB'

    input:
    val(sample)

    output:
    tuple val(sample), path("${sample}.fastq.gz")

    script:
    """
    fastq-dump ${sample} > ${sample}.fastq.gz
    """
}
```

## Anatomy of a process

```
process GET_SRA_BY_ACCESSION {

    cpus 2
    memory '8 GB'

    conda 'sra-tools=2.11.0'
    container 'ncbi/sra-tools:2.11.0'

    input:
    val(sample)

    output:
    tuple val(sample), path("${sample}.fastq.gz")

    script:
    """
    fastq-dump ${sample} > ${sample}.fastq.gz
    """
}
```

# Anatomy of a process

```
process GET_SRA_BY_ACCESSION {

    cpus 2
    memory '8 GB'

    conda 'sra-tools=2.11.0'
    container 'ncbi/sra-tools:2.11.0'

    input:
    val(sample)

    output:
    tuple val(sample), path("${sample}.fastq.gz")

    script:
    """
    fastq-dump ${sample} -X {params.depth} > ${sample}.fastq.gz
    """
}
```

## Anatomy of a workflow

```
workflow {

    // Define SRA input data channel
    ch_sra_ids = Channel.fromList ( ["SRR935090", "SRR935091", "SRR935092"] )

    // Define the workflow
    GET_SRA_BY_ACCESSION (
        ch_sra_ids
    )
}
```

## Anatomy of a workflow

```
workflow {

    // Define SRA input data channel
    ch_sra_ids = Channel.fromList ( ["SRR935090", "SRR935091", "SRR935092"] )

    // Define the workflow
    GET_SRA_BY_ACCESSION (
        ch_sra_ids
    )
    RUN_FASTQC (
        GET_SRA_BY_ACCESSION.out
    )
}
```

# Anatomy of a workflow

```
workflow {

    // Define SRA input data channel
    ch_sra_ids = Channel.fromList ( ["SRR935090", "SRR935091", "SRR935092"] )

    // Define the workflow
    GET_SRA_BY_ACCESSION (
        ch_sra_ids
    )
    RUN_FASTQC (
        GET_SRA_BY_ACCESSION.out
    )
    RUN_MULTIQC (
        RUN_FASTQC.out.collect()
    )
}
```

## Executing Nextflow

```
# Execute a workflow
$ nextflow run main.nf
```

```
# Re-run using cached results
$ nextflow run main.nf -resume
```

```
# Execute with a specific configuration file
$ nextflow run main.nf -c nextflow.config
```

```
# Supply a custom parameter
$ nextflow run main.nf --my_param "my value"
```

```
# Use Docker or Singularity
$ nextflow run main.nf -with-docker
$ nextflow run main.nf -with-singularity
```

```
# Use a pre-defined configuration profile
$ nextflow run main.nf -profile uppmax
```

## Differences between Snakemake and Nextflow

| | Snakemake | Nextflow |
|---|---|---|
| Language | Python | Groovy |
| Data | Everything is a file | Can use both files and values |
| Execution | Working directory | Each job in its own directory |
| Philosophy | "Pull" | "Push" |
| Dry-runs | Yes | No |
| Track code changes | No | Yes |

- Question: But, which one is the best?
- Answer: Both - it's mostly up to personal preference!

Questions?