An Example Snakemake Workflow

SciLifeLab

NBIS

The GenErode pipeline

- Developed in a NBIS project with Love Dalén's lab (Centre for Palaeogenetics, SU & NRM)
- Compares population genomics statistics from historical and modern samples of endangered populations

Sumatran rhinoceros (Dicerorhinus sumatrensis), critically endangered

- Data processing from fastq files to BAM & VCF files plus downstream population genomics analyses

The GenErode pipeline

- Started at Snakemake version 3.10 (!), current pipeline runs with Snakemake version 5.22
- Historical and modern samples are processed in parallel
- Whole-genome resequencing data from historical/ancient samples needs special processing as DNA degrades over time
- Some analyses or filtering steps are run separately for modern and historical samples, or only for historical samples

Analysis Tracks of the Workflow

- Data processing track
  - Repeat element identification
  - Fastq file processing
    - Optional: mapping to mitochondrial genomes
  - Mapping to reference genome
  - BAM file processing
    - Optional: base quality rescaling for historical samples
    - Optional: subsampling to target depth
  - Genotyping
    - Optional: CpG site identification & removal
  - VCF file processing & merging per dataset

Analysis Tracks of the Workflow

- BAM file track
  - Optional: mlRho
- VCF file track
  - Optional: plink (PCA)
  - Optional: plink (runs of homozygosity)
  - Optional: snpEff
  - Optional: GERP++ & relative mutational load calculation

The Workflow Structure

- Rules with the actual analyses in separate Snakefiles (in `workflow/rules/`)

- `Snakefile`
  - Python code to create sample and readgroup ID dictionaries & lists from metadata tables and a config file
  - `include` of rule Snakefiles
  - `all` rule collecting output files produced by the different rule Snakefiles

- UPPMAX / slurm system:
  - `cluster.yaml` file to set up slurm profile

The Workflow Structure

- Metadata files for historical and modern samples (separately)
  - Sample IDs, readgroup IDs, sequencing technology, paths to fastq files

- Example `historical_samples.txt` file (whitespace-separated)

samplename_index_lane	readgroup_id	readgroup_platform	path_to_R1_fastq	path_to_R2_fastq
VK01_01_L2	BHYOX3ALTH.L2.01	illumina	data/S1/P01_2.R1.fq.gz	data/S1/P01_2.R2.fq.gz
VK01_02_L2	BHYOX3ALTH.L2.02	illumina	data/S1/P02_2.R1.fq.gz	data/S1/P02_2.R2.fq.gz

The Workflow Structure

- Config file `config.yaml` (to be edited by users, placed in `config/`)
  - Paths to input data and metadata tables
  - Selection of analysis steps to be run
  - Parameters for different rules
  - Lists with samples for optional analyses

```
#####################################################################
# 1) Full path to reference genome assembly.
# Reference genome has to be checked for short and concise fasta
# headers without special characters.
# File name extension can be ".fasta" or ".fa".
ref_path: ""
#####################################################################


#####################################################################
# 2) Relative paths (from the main snakemake directory) to metadata tables of samples.
# Example files can be found in "config/"
historical_samples: "" # leave empty ("") if not run for historical samples.
modern_samples: "" # leave empty ("") if not run for modern samples.
#####################################################################
```

How to choose which steps to run

Step 1: Use booleans in the config file (`config/config.yaml`) as on/off switches

```
#####
# FastQC on raw reads, adapter trimming, read merging (historical samples), FastQC on trimmed reads.
fastq_processing: True

[...]
#####

#####
# Map historical and modern reads to reference genome assembly (specified above).
mapping: False
#####
```

How to choose which steps to run

Step 1: Use booleans in the config file (`config/config.yaml`) as on/off switches

- For many analysis steps, parameters can be specified in the config file:

```
#####
# FastQC on raw reads, adapter trimming, read merging (historical samples), FastQC on trimmed reads.
fastq_processing: True

# Adapter sequences for trimming of historical samples using SeqPrep v1.1 (modified source code)
# Forward read primer/adapter sequence to trim historical samples in SeqPrep v1.1 (parameter "-A")
hist_F_adapter: "AGATCGGAAGAGCACACGTCTGAACTCCAGTCACNNNNNNNNATCTCGTATGCCGTCTTCTGCTTG" # example from Meyer & Kircher 2010
# Reverse read primer/adapter sequence to trim historical samples in SeqPrep v1.1 (parameter "-B")
# When using double indices, include full adapter length (replacing the index by "N")
hist_R_adapter: "AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGTAGATCTCGGTGGTCGCCGTATCATTT" # example from Meyer & Kircher 2010

[...]

# Minimum read length allowed after trimming.
# Historical samples (SeqPrep v1.1 with modified source code)
hist_readlength: "30" # default setting: 30 bp

# Modern samples (trim-galore)
mod_readlength: "30"
#####
```

- These parameters can be used in the workflow with the syntax `config["parameter_name"]`

How to choose which steps to run

Step 2: Use some Python code, `include` and the rule `all` to figure out what the workflow will do

- The main `Snakefile` contains an empty Python list `all_outputs` to collect output files from the included rule Snakefiles,

```
all_outputs = []
```

- the `include` variable to attach the rule Snakefiles corresponding to the analysis steps that were set to `True` in the config file,

```
if config["fastq_processing"]:
    include: "workflow/rules/1.1_fastq_processing.smk"
```

- and the rule `all` that takes the output files from the list `all_outputs` as input

```
rule all:
    input: all_outputs
```

How to choose which steps to run

Step 2: Use some Python code, `include` and the rule `all` to figure out what the workflow will do

- The rule Snakefile (`workflow/rules/1.1_fastq_processing.smk`) contains some Python code to add its final output files to the list `all_outputs` in the main `Snakefile`

```
import os
if os.path.exists(config["historical_samples"]):
    all_outputs.append("data/raw_reads_symlinks/historical/stats/multiqc/multiqc_report.html")
    all_outputs.append("results/historical/trimmed_merged_reads/stats/multiqc/multiqc_report.html")

if os.path.exists(config["modern_samples"]):
    all_outputs.append("data/raw_reads_symlinks/modern/stats/multiqc/multiqc_report.html")
    all_outputs.append("results/modern/trimmed_reads/stats/multiqc/multiqc_report.html")
```

- By using the `if` statement to check for the presence of historical or modern metadata files, the workflow can also be run only for historical or only for modern samples

Questions?

SciLifeLab

NBIS

SciLifeLab

NBIS