A Software Workflow for Automated Analysis of Genome (Re-)Sequencing Projects

A LABORATORY PROTOCOL

Plant Breeding and Genetics Laboratory, Seibersdorf, Austria

Version 2_10_05_2022

CONTENT

1 EXECUTIVE SUMMARY

DNA sequencing cost dropped sharply during the past decade. Current low prices enable large-scale sequencing of entire crop genomes and identifying virtually all DNA variation. This enables genomics approaches to plant (mutation) breeding.

Genomics approaches have the potential to significantly shorten crop breeding cycles because they can make the process more direct, predictive, and less dependent on the growing seasons. The efficiency gain comes from linking desired crop traits to the underlying, heritable DNA variation, and subsequently tracking the causal DNA variants through the breeding process in lieu of the trait, i.e., by Marker Assisted Breeding.

Despite their disruptive impact on plant mutation breeding, the adoption of genomics approaches by Member States has been slow. PBGL identified the challenges of data analysis as one obstacle to adoption. To address this issue, PBGL developed a software workflow that significantly reduces the complexity of DNA sequence analyses.

The currently most popular, because cost-effective DNA sequencing technology in plant breeding applications is 2nd-generation sequencing with Illumina and related technologies. They produce hundreds of millions of short sequencing reads per sequencing run. Handling the high-volumes of raw, short-read DNA sequencing data and extracting the relevant information is technically challenging. PBGL's software workflow greatly simplifies this process and, flanked by training courses where necessary, should empower researchers in Member States to embrace genomics technologies.

The workflow is available online, free and open source, ready for use by Member States. We provide documentation with detailed instructions. Researchers with computer programming expertise will not need any training. A targeted training course of one to two weeks should be sufficient to teach the use of the workflow to non-experts.

Note for use:

This protocol is made available for use as working material and is not an official IAEA publication. Although great care has been taken to maintain the accuracy of information contained in this protocol, neither the IAEA nor its Member States assume any responsibility for consequences which may arise from its use. As the material is still under development it may be updated in the future and published in final form. Please check the iaea.org website for updates.

Contributors to this working draft –

WARTHMANN, N., PBGL (Text),

MORALES ZAMBRANA, A. E., PBGL – (Illustrations)

2 BACKGROUND

The Motivation: Mainstreaming Genomics Approaches

Over the past decade, the economics of sequencing DNA has radically changed. With advent of the so-called Next Generation Sequencing technologies (NGS), prices have dropped to a point where researchers around the world can sequence entire genomes. In plant breeding, this information is used to catalogue genetic variation and identify advantageous alleles important for improving our crops, for example, by genetic mapping. These alleles can then be tracked through the breeding process with molecular (DNA) markers developed from linked variants.

This is significant. Breeding relies on heritable differences between crop varieties but has, for the longest time, been restricted to evaluating phenotypes. Being able to efficiently associate desired phenotypes -the traits- with differences in the underlying genes accelerates the breeding process and shortens breeding cycles. Genomics-assisted breeding makes the breeding process more direct, predictive, and less dependent on growing seasons.

Despite their significance for plant breeding and the rather cheap cost of DNA sequencing, genomic approaches are not yet mainstream. To the contrary, their efficient application has so far been restricted to advanced research institutions. Wide-spread adoption is slow; too slow. More wide-spread adoption of genomics approaches is necessary for accelerating plant breeding and crop improvement progress worldwide. Plant breeders around the world in their respective institutions must be empowered to unleash genomic tools in their work addressing production constraints in a plethora of crops in their respective diverse geographies, environments, and cultural management systems.

Mainstreaming genomics approaches requires lowering the barrier of entry. While DNA sequencing capacity is readily accessible from around the world, the subsequent data analysis is challenging and still requires bioinformatics expertise and sophisticated hardware. Outsourcing options exist but are usually expensive, difficult to manage, and often don’t meet expectations. Ideally, researchers in Member State institutions remain in control of the entire process and have the capacity to analyse, use, and re-analyse their data themselves.

PBGL has been using NGS-enabled genomics approaches for a several years now. We understand the challenges and have automated the initial analysis process. We are making our analysis pipeline in form of a software workflow publicly available, free and open source. This document explains the concept and is the technical documentation to the software workflow. The use is described in enough detail that researchers with computer programming expertise in Linux/Unix environments will be able to conduct analyses following the instructions; knowledge of python will be an advantage but note that particular Bioinformatics expertise is not required.

The Challenges of Genomics Approaches

Genomics approaches generally comprise the steps:

1) **Experimental Design,**
2) **Sample Preparation,**
3) **DNA Sequencing,**
4) **Data Analysis.**
    a. **Initial Analysis**
    b. **Downstream Analysis**

The software workflow in this protocol conducts the Initial Analysis (4a)

The **DNA Sequencing (3)** itself is easily outsourced to commercial providers. These services are accessible from around the globe as samples are easily shipped and data is digitally transferred.

The required **Sample Preparation (2)** prior to sequencing is sufficiently standardized and can either also be outsourced to the sequencing service provider or performed by the plant breeder. The sample preparation involves molecular biology techniques slightly more advanced than DNA extraction, and plant breeders will make their decision based on capacity and cost. For only a handful of samples it might be more cost-effective to outsource the entire process and ship DNA, but for dozens to hundreds of samples it can be better to perform the sample preparation on site and ship so-called sequencing libraries, i.e., DNA in a form that is ready to be put on the DNA sequencer.

Finally, for **Data Analysis (4)** it is helpful to distinguish two steps:

- **Initial Analysis (4a)**, which reduces the raw sequencing data to the relevant information on DNA variation, and
- **Downstream Analysis (4b)** that then draws conclusions from the sequencing information in context of the **Experimental Design (1)**.

Those are different types of analyses with very different demands on capacity with respect to hardware, software, and skills of the respective operator/analyst.

Downstream Analysis (4b) and **Experimental Design (1)** go hand in hand. Both have been and remain the responsibility of the plant breeder/geneticist. The concepts have not changed, and most plant breeders are adequately trained. What has changed is the ease and speed of sequence data generation, and the unprecedented resolution of the data allowing for whole-genome views.

Required capacity building in this area will focus on educating plant breeders about the options available and accessible to them, inspiring a sense of the new possibilities for experimental design and teaching a few procedures on how to work with and visualise the data summaries derived from the sequencing data.

Generating those summaries, however, is the task of the **Initial Analysis (4a).** Initial analysis describes the process of turning the raw sequence data that comes from the 2nd generation

sequencing machine (hundreds of millions of DNA sequencing reads) into information relevant to the plant breeder.

The concept of the **Initial Analysis** is not difficult to understand, but the technical difficulties are such that it can currently only be performed by bioinformatics experts. Reasons are that the amount of data is large, the analysis comprises many subsequent interdependent steps, the required software packages are written in a variety of different programming languages, and individual projects are sufficiently different that every analysis is slightly different and the analysis will need adopting.

Our Solution: Reducing the Complexities

Recently, modern tools became available that can be used to abstract some of the computer science complexities from the user. Of interest in our context are virtual environments with associated software repositories (e.g., conda), and workflow management systems (e.g., CWL, snakemake, nextflow). With those tools in hand, it becomes possible to mainstream even complex analyses.

With workflow management systems the desired analyses can be cast into series of pre-defined interconnected steps. The user is then only concerned with providing the input, configuring the steps, and collecting the output, but doesn't need to be involved with any intermediate step.

In addition, the use of virtual environments and associated software repositories makes traditional software installations obsolete. A virtual environment can be viewed as an independent capsule within a computer. It is virtual, configured to contain all software that the workflow requires, and can be generated within minutes on any hardware, automatically and even on-the-fly.

The workflow management system scales the analysis workflow to optimally use the provided hardware –server, cluster, grid or cloud environments without the need to modify the workflow definition.

Note that, since all steps and the software are predefined, the analyses are automatically documented through configuration files in sufficient detail that they can be re-run by anyone, anywhere, and the result will be the same. This addresses the issue of reproducibility, a frequent complaint in biological sciences today.

3 WORKFLOW IMPLEMENTATION

This workflow capitalizes on recent developments in computer science, namely virtual environments and workflow management systems. The workflow conducts **Initial Analysis** to extract genetic variant information from raw 2nd-generation DNA sequencing reads as they are produced by short-read DNA sequencing machines. All analysis software as well as the workflow software is free and open source.

We chose *Snakemake* as workflow management system. *Snakemake* is based on *Python*, a very popular general purpose programming language. For software and virtual environment management we chose *conda*. Installations of git and conda are the only prerequisites. All other software will be installed through git and conda as is explained further below.

Main analysis steps and respective software:

Analysis step	**Software**
Prepare/clip the raw reads	AdapterRemoval
K-mer clustering	kWIP and/or mash
Read Alignment to the reference genome	bwa mem and/or NextGenMap (ngm)
Mark Duplicates	samtools markdup
Realign indels	abra2
Call Variants	freebayes and/or mpileup
Filter Variants	bcftools view
Annotate Variants	snpEff

Important features of our workflow are the possibility of combinatorial use of read aligners with variant callers and the option of having more than one reference genome. The user can conduct an analysis with several different combinations of tools against several reference genomes at the same time and later evaluate which results are best, and best suited for their downstream analyses. Genomes are complex, and reference genome assemblies vary greatly in completeness and quality. Different tools have different strengths and weaknesses. Using different tools can improve the overall result.

All software components are publicly available on GitHub:

Software	**GitHub Repository**
AdapterRemoval	https://github.com/MikkelSchubert/adapterremoval
kWIP	https://github.com/kdm9/kWIP
mash	https://github.com/marbl/Mash
bwa mem	https://github.com/lh3/bwa
NextGenMap (ngm)	https://github.com/Cibiv/NextGenMap
abra2	https://github.com/mozack/abra2
samtools/bcftools	https://github.com/samtools
SNPeff	https://github.com/pcingola/SnpEff

4 AVAILABILITY OF THE WORKFLOW AND DOCUMENTATION

The Snakemake Workflow software is hosted on **github.com**
https://github.com/pbgl/dna-proto-workflow

Detailed technical documentation in online at **read-the-docs.io**
https://dna-proto-workflow-master.readthedocs.io/en/latest/

Disclaimer

5 LITERATURE

Snakemake, introduction and documentation: https://snakemake.readthedocs.io/en/stable/

Conda, documentation: https://docs.conda.io/en/latest/, https://bioconda.github.io

Problem statement and choice of analysis tools:

- Sboner A, Mu XJ, Greenbaum D, Auerbach RK, Gerstein MB. **The real cost of sequencing: higher than you think!** *Genome Biology 2009 10:R88. 2011 Aug 25;12(8):125.*
- Stephens ZD, Lee SY, Faghri F, Campbell RH, Zhai C, Efron MJ, et al. **Big Data: Astronomical or Genomical?** *PLoS Biol. 2015 Jul;13(7):e1002195.*
- Yao Z, You FM, N'Diaye A, Knox RE, McCartney C, Hiebert CW, et al. **Evaluation of variant calling tools for large plant genome re-sequencing**. *BMC Bioinformatics. 2020 Aug 17;21(1):360.*

Bibliography of the individual analysis tools:

AdapterRemoval	Schubert, Lindgreen, and Orlando (2016). **AdapterRemoval v2: rapid adapter trimming, identification, and read merging**. BMC Research Notes, 12;9(1):88.
kWIP	Murray KD, Webers C, Ong CS, Borevitz J, Warthmann N (2017) **kWIP: The k-mer weighted inner product, a de novo estimator of genetic similarity**. PLoS Comput Biol 13(9): e1005727.
mash	Ondov BD, Treangen TJ, Melsted P, Mallonee AB, Bergman NH, Koren S, et al. **Mash: Fast Genome and Metagenome Distance Estimation Using MinHash**. Genome Biology. 2016;17:132
bwa mem	Li H. (2013) **Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM**. arXiv:1303.3997v2
NextGenMap (ngm)	Fritz J. Sedlazeck, Philipp Rescheneder, and Arndt von Haeseler. **NextGenMap: fast and accurate read mapping in highly polymorphic genomes**. Bioinformatics (2013) 29 (21): 2790-2791
abra2	Lisle E Mose, Charles M Perou, Joel S Parker**. Improved indel detection in DNA and RNA via realignment with ABRA2**. Bioinformatics, 35(17), 2966–2973.
freebayes	Garrison E, Marth G. **Haplotype-based variant detection from short-read sequencing**. arXiv preprint arXiv:1207.3907 [q-bio.GN] 2012
samtools/bcftools	Petr Danecek, James K Bonfield, Jennifer Liddle, John Marshall, Valeriu Ohan, Martin O Pollard, Andrew Whitwham, Thomas Keane, Shane A McCarthy, Robert M Davies, Heng Li. **Twelve years of SAMtools and BCFtools**. Gigascience (2021) 10(2):giab008
SNPeff	Cingolani P, Platts A, Wang le L, Coon M, Nguyen T, Wang L, Land SJ, Lu X, Ruden DM. **A program for annotating and predicting the effects of single nucleotide polymorphisms, SnpEff: SNPs in the genome of Drosophila melanogaster strain w1118; iso-2; iso-3**. Fly. 2012 Apr-Jun;6(2):80-92.

6 WORKFLOW TECHNICAL DOCUMENTATION

6.1 OVERVIEW AND PREREQUISITES

This collection of Snakemake rules constitutes a workflow for the automated analysis of high-throughput DNA sequencing data. We use it mostly for genome re-sequencing data produced with Illumina-type sequencing instruments. The workflow is designed to run the subsequent steps of the entire analyses automatically.

6.1.1 The (brief) Principle of the Workflow Language Snakemake

In Snakemake, analysis workflows are defined as a series of rules. In our case, rules define the analysis steps. Each rule has input and output files. Calling a rule will invoke all necessary upstream rules. It is hence generally sufficient to request the desired output file(s) of the downstream most rule and the entire workflow will be executed. In practice, users will provide input files, configure project meta information, adjust configuration parameters, and execute `snakemake`. The workflow will then automatically perform all the (necessary) analysis steps to produce the defined output.

In general, a rule will be run if:

- Snakemake realizes that the expected output files of a rule are not (yet) present, or
- Snakemake realizes that the expected input files or configuration parameters of a rule have changed since the last run.

Hence, when analyses are re-run, existing and valid intermediate output files from a previous run are re-used, saving precious time and resources. For details, please consult the Snakemake documentation. For the different use cases in this workflow, invoke the rule that produces the desired output.

6.1.2 Analysis Workflow Use Cases

For this particular workflow we distinguish two different use cases: **De-novo** and **variant calling** .

6.1.2.1 "De-novo"

Choosing this option will conduct a reference-free comparison of *samples* based on the raw sequencing reads using k-mers. The workflow invokes the software tools kWIP and/or mash and outputs distance matrices and PCA plots. A flowchart illustrating the summarised workflow can be found in this document under section 7.1. The workflow for this use case consists of the following steps:

#	**Task**	**Rule**	**softwares**
1	Prepare/clip the raw reads	readQC	AdapterRemoval
2	Calculate distances	denovo	kWIP and/or mash
3	Perform PCA and plot	denovo	Utils/R-Scripts

- Invoke this option by running the rule: `rules.denovo.input`

Required input for `rules.denovo.input` are fastq files and the workflow will return distance matrices between *samples* produced by kWIP and/or mash.

> NOTE: We recommend performing such de-novo analysis for every project. Clustering at the level of sequencing runs can be used to confirm metadata and detect mixups.

6.1.2.2 "Variant Calling"

Choosing this option will run a full re-sequencing analysis ending in filtered bcf/vcf files. It detects variants and genotypes in *sets* of *samples* based on the alignments of the sequencing reads against one or several user-defined reference genome(s). Reads can be aligned with bwa and/or NextGenMap (ngm), and variants can be called with freebayes and/or mpileup. Between read alignments and variant calling, PCR duplicates are marked (with samtools markdup) and indels are realigned (using abra2). If reference genome annotation is provided, the effects of variants on gene integrity can also be predicted using the software snpEff. A flowchart illustrating the summarised workflow can be found in this document under section 7.2.

The full workflow for this use case consists of the following steps:

#	**Task**	**Rules**	**softwares**
1	Prepare/clip the raw reads	ReadQC	AdapterRemoval
2	Align the reads to the reference genome	align	bwa and/or ngm
3	Mark Duplicates	align	samtools fixmate samtools sort samtools markdup
4	Realign indels	align	samtools merge abra2
5	Call Variants	varcall	freebayes and/or mpileup
6	Filter Variants	varcall	bcftools view
7	Annotate Variants	annotate	snpEff

This option can be invoked in 2 ways:

- selecting `rules.varcall.input` will result in one or several **vcf files**, one for each combination of sample-set, reference genome, read aligner, variant caller, and variant filter.
- selecting the rule `rules.annotate.input` will result in one or several **annotated vcf files** and additional summaries; running `rules.annotate.input` is only meaningful when a genome annotations is available and provided in form of a snpEff database. Currently, only one reference genome annotation can be provided at a time. Hence, invoking the variant calling workflow through `rules.annotate.input`, will restrict the workflow upstream to only one reference genome.

Required input files are fastq files and a genome reference sequence(s) (fasta). The rule `snpeff` in addition depends on a genome annotation in form of a snpEff database matching the reference genome. For maximum flexibility and ease of troubleshooting we recommend to first run the re-sequencing analysis by invoking `rules.varcall.input`, and upon successful completion invoke the workflow again, this time selecting/uncommenting `rules.annotate.input`.

6.1.2.3 "Annotate"

Choosing the option `rules.annotate.input` will annotate bcf/vcf files found in **output/variants/final/** and write annotated vcf.gz files to **output/variants_annotated/**. This analysis has only one step:

#	**Task**	**Rules**	**softwares**
1	Annotate Variants	annotate	snpEff

Typically, `rules.annotate.input` is run after a completed run of `rules.varcall.input`. A snpEff run will complete within a matter of minutes. It will run on files specified through entries in the snpeff section of the `config.yml` file:

- `config['snpeff']['ref']` one chosen reference genome,
- `config['snpeff']['database']` one corresponding snpEff database,
- and one or several of `['samplesets']`, `['aligners']`, `['callers']`, and `['filters']`

that together specify the files to annotate.

6.1.3 Hardware Requirements

The workflow is parallelised and Snakemake will make efficient use of available resources on local machines as well as on compute clusters. It will run faster the more resources are available, but it performs fine on smaller machines. For routine applications we have used the workflow on a budged workstation, HP Z820 with 32 cores, 64 GB RAM running Ubuntu 16.04, and on a Virtual Machine in the cloud, AZURE with 16 cores and 512 GB RAM running Ubuntu 18.04.

Snakemake allows for fine-tuning resource allocation to the individual rules, i.e., number of processors and memory. We configured each rule with reasonable defaults, but they can be tailored to your particular size project and hardware. For details please consult the Snakemake documentation. In general though, if limited by memory, do not parallelise too aggressively.

6.1.4 Software Dependencies

We recommend running the workflow in its own `conda environment` on a Linux Server. Dependencies are listed in `envs/condaenv.yml` and `envs/additional.yml`. A brief explanation how to use these files to generate the conda environment is further below. For comprehensive

explanation please consult the conda documentation. For software that is not available through conda on some important platforms we make the specific binaries available in `envs/`; currently mainly abra2.jar.

6.2 WORKFLOW USE CHECKLIST

Steps:

1. Create a new github repository in your github account using this workflow as a template
2. Clone your newly created repository to your local system where you want to perform the analysis
3. Create and activate the conda environment
4. Provide reference genome(s) and annotation(s) in `/genomes and annotation/`
5. Specify the locations of input files and their meta data in `/metadata/sample2runlib.csv`
6. Provide lists of *samples* as sets to analyse in `metadata/samplesets/`
7. Uncomment the respective workflow option for your use case in the `Snakefile`
8. Configure software parameters in `config.yml`
9. Adapt `snpeff.config` (Optional in case the `snpeff` rule will be called)
10. Run `snakemake`

For standard applications no additional edits are necessary. The rules reside in `rules/*.smk`. Most rules have explicit shell commands with transparent flag settings. Expert users can change these for additional control.

6.3 WORKFLOW USE IN DETAIL

We recommend using Linux, managing the software dependencies trough conda/mamba and running the workflow in a dedicated conda virtual environment. The virtual environment is created, activated, and then all software will be available.

6.3.1 Creating the Virtual Environment (conda)

The preferred way to create the environment is with `mamba`. The code below will install mamba (if not already available), create a virtual environment named "dna-proto", and activate it. If you are new to conda then please consult the conda documentation to get started.

```
$ conda install mamba
$ mamba env create --file env/all-dependencies.yml
$ conda activate dna-proto
```

Alternatively (and only if the above doesn't work as intended), create the empty environment and then `conda install` (or `mamda install`) the individual programs manually. They are listed in `env/all-dependencies.yml.` Specify the correct channel and software version where required. Below we give an example, but please consult the conda (and bioconda) documentation.

Example:

```
$ conda create -- name dna-proto
$ conda install -c bioconda samtools=1.9
Etc.
```

In case of manual installs, it is convenient to add all required channels to the `conda config`. You will need below channels:

```
$ conda config --show channels
channels:
      - defaults
      - bioconda
      - conda-forge
      - r
$ conda config --add channels bioconda
$ conda install samtools=1.9
```

The required channels are listed in the respective `*.yml` files. Configuring channels has the pitfall of rare ambiguities and collisions. Please consult the conda documentation for "managing channels".

Once set up, don't forget to activate your conda environment:

```
$ conda activate dna-proto
```

6.3.2 Reference Genome

The workflow will look for the reference genome(s) and its associated files in `genomes_and_annotations/`. We provide an example directory tree in `genomes_annotations/readme`. We recommend creating a separate directory for each reference genome. They can be softlinks. Each reference genome directory must contain the necessary assembly file (.fa or .fna) and the associated files required by the aligners. Generate those files in this directory from the assembly file (fasta: .fa or .fna) like so:

```
$ samtools faidx <reference-genome.fa>
$ bwa index -a bwtsw <reference-genome.fa>
$ ngm -r <reference-genome.fa>
```

directory tree of the reference genome directory once fully prepared

```
~/genomes_and_annotations/
├── GCF_004118075.1_ASM411807v1
    ├── GCF_004118075.1_ASM411807v1_genomic.fna
    ├── GCF_004118075.1_ASM411807v1_genomic.fna.amb
```

```
    ├── GCF_004118075.1_ASM411807v1_genomic.fna.ann
    ├── GCF_004118075.1_ASM411807v1_genomic.fna.bwt
    ├── GCF_004118075.1_ASM411807v1_genomic.fna-enc.2.ngm
    ├── GCF_004118075.1_ASM411807v1_genomic.fna.fai
    ├── GCF_004118075.1_ASM411807v1_genomic.fna-ht-13-2.3.ngm
    ├── GCF_004118075.1_ASM411807v1_genomic.fna.pac
    └── GCF_004118075.1_ASM411807v1_genomic.fna.sa
```

6.3.3 Reference Genome Annotation

In order to annotate variant effects using snpEff you will need a snpEff database (`snpEffectPredictor.bin`) for the relevant reference genome. The location of the `snpeff` database is configured in `snpeff.config`. It is currently set to `genomes_and_annotations/snpeffdata/` and again, we recommend separate subdirectories for the reference genomes also in this directory. Appending "_snpeff" to these directory names will help avoid confusion. In order to create a `snpeff` database, this `<reference_genome>_snpeff` directory must contain the reference genome and the annotation in files named `sequences.fa` and `genes.gff`; they again can be soft links. (See `genomes_and_annotations/snpeffdata/readme`.) For building the database you will need to add the respective entry in `snpeff.config` and then, from the root directory of the workflow (the directory that contains the `snpeff.config` file), execute:

```
$ snpEff build –gff3 genomes_and_annotations/snpeffdata/<reference-genome>_snpeff
```

While a `.gtf` file can also be used (-gtf 22), we have better experience building databases from `.gff` files. For a detailed explanation of the `snpeff` build process please consult the snpEff documentation.

Below is an example directory tree of `genomes_and_annotations/` for a cowpea reference genome downloaded from NCBI. Notice that we store the reference genomes elsewhere and use soft links. Compare also to our entries in `snpeff.config` under "Non-standard Databases" and replicate accordingly. The database `snpEffectPredictor.bin` will be generated by `snpEff build`.

```
genomes_and_annotations/
├── GCF_004118075.1_ASM411807v1 -> ~/genomes/Cowpea/
├── readme
└── snpeffdata
    └── GCF_004118075.1_ASM411807v1_snpeff
        ├── genes.gff -> ~/genomes/Cowpea/GCF_004118075.1_ASM411807v1_genomic.gff
        ├── genes.gtf -> ~/genomes/Cowpea/GCF_004118075.1_ASM411807v1_genomic.gtf
        ├── protein.fa -> ~/genomes/Cowpea/GCF_004118075.1_ASM411807v1_protein.faa
        ├── sequences.fa -> ~/genomes/Cowpea/GCF_004118075.1_ASM411807v1_genomic.fna
        └── snpEffectPredictor.bin
```

6.3.4 *Providing Required Meta Information*

6.3.4.1 *Samplesets*

We use the term *sample* as the entity of interest. Our workflows call variants on *samples* within one *set*. Lists of sample names must be provided as text file (`.txt`) in `/metadata/samplesets/`. Each file defines a set. The `samplesets:` in `config.yml` refer to those files and must match the filenames. We implemented a glob: *all_samples*, which will have the effect of concatenating all `*.txt` files in `/metadata/samplesets/` into an additional set comprising all samples across all sets. The intention is to enable easy addition or removal of samples to/from an existing analysis.

6.3.4.2 *Sample Definitions*

In practice, the unit of interest oftentimes is the individual (plant). DNA is extracted from an individual and turned into one or several sequencing libraries that are then sequenced in one or several sequencing runs. In order to compare individuals all respective fastq files need to be assigned to the same *sample*. For our workflow, the run - library - sample relationships need to be defined in `/metadata/sample2runlib.csv` making explicit which fastq files constitute the samples. The entries in columns *run* and *library* are together used as the primary key for the **sequencing run** and thus the fastq file(s). The user must make sure that any *run – library* combination is unique. **Sequencing runs** are assigned to the same *sample* through an identical entry in the *sample* column. fastq files can be provided either as separate forward and reverse read files (*fq1*, *fq2*) or as interleaved fastq (*il_fq*), with the respective other column(s) empty. Within one `sample2runlib.csv` file interleaved, and two-file input can be mixed.

Example `sample2runlib.csv` file:

run	Library	sample	fq_1	fq_2	il_fq
Run1	A-500bp	A	<path to file>	<path to file>	
Run1	A-300bp	A	<path to file>	<path to file>	
Run2	A-500bp	A	<path to file>	<path to file>	
Run2	B-300b p	B			<path to file>
Run2	B-500bp	B			<path to file>
Run1	C-500bp	C	<path to file>	<path to file>	

6.3.4.3 *Regions of Interest for Variant Calling*

Variant calling can be restricted to particular regions of interest through `metadata/contigs_of_interest.bed`: A file in bed file format listing identifier, start-, and end-positions (tab delimited, no header). This is helpful for exome capture data but also to restrict the analysis to specific chromosomes. Genome assemblies often contain the main chromosomes and in addition many orphan fragments that often are not of interest. Below example will restrict variant calling to the 11 chromosomes plus the chloroplast of cowpea. The hundreds of additional contigs in the cowpea reference genome are available for read

mapping, but variants will not be called for them and their variants will subsequently not be present in the bcf/vcf files. Note that lines starting with '#' will be disregarded.

Example `contigs_of_interest.bed` file:

```
NC_040279.1	0	42129361
NC_040280.1	0	33908088
NC_040281.1	0	65292630
NC_040282.1	0	42731077
NC_040283.1	0	48746289
NC_040284.1	0	34463471
NC_040285.1	0	40876636
NC_040286.1	0	38363498
NC_040287.1	0	43933251
NC_040288.1	0	41327797
NC_040289.1	0	41684185
NC_018051.1	0	152415
```

6.3.5 Workflow Configuration

6.3.5.1 config.yml

Central place for the user to configure the workflow behaviour and software parameters is `config.yml`. There are comments in the file that explain the configuration parameters and options.

In Snakemake, calling a rule will trigger running of the upstream rules. When editing `(/config.yml)` it is important to only configure the intended most downstream rule (`varcall:`, `annotate:`, or `denovo:`). These settings will be propagated upstream. `qc:` is independent and will require editing, particularly the `_DEFAULT_` adaptors used. The standard adaptors for Truseq- and Nextera-Libraries are given for reference, paste under `_DEFAULT_` as required.

An important configuration parameter is the location of a suitable temporary directory ("tmp/"). Several rules make extensive use of the "tmp/" directory to temporarily store large files. Oftentimes, standard home directories on compute servers or cluster nodes are too small. Central place for the configuration of the "tmp/" directory is currently under the abra2 configuration options (`abra2:temp:`).

6.3.5.2 Additional Configuration

Expert users can change the rules themselves by editing `rules/*.rules.smk`. Use caution! We have chosen reasonable default settings and recommend modifying rules only when you know what you are doing. When allocating more cores to rules, pay attention that some rules are very memory intensive and some shell commands are piped and are in fact using more than 1 core per process.

6.3.6 *Running the Workflow*

To run the workflow, un-comment the respective rule for the desired use case in the `Snakefile` and run `snakemake`.

```
$ snakemake –npr
$ snakemake –j 6 --no-temp -kpr
```

For details on command line options for snakemake please consult the Snakemake documentation. Un-comment only the one most downstream rule for your use case. Currently, these use case rules are:

```
# USER OPTIONS
#       rules.denovo.input,
#       rules.varcall.input,
#       rules.annotate.input,
```

A rule that encounters missing input files will invoke the respective upstream rule(s). E.g., when `rules.annotate.input` is uncommented and snakemake is run for the first time, the entire workflow from `readqc` (= adapter and quality clipping), `align` (= read alignments, duplicate marking, indel realignment), `varcall` (= variant calling), and `annotate` (= variant functional annotation) will run in one go. In case no `snpeff` database is supplied, then rule `rules.annotate.input` cannot be run. Use `rules.varcall.input` instead.

When configuring `config.yml`, keep in mind that configuration parameters of a downstream rule take precedence because parameters have to propagate upstream. I.e., When running `varcall`, the user must set the alignment parameters in the `varcall:` section; same for the `annotate:` parameters; they must be set under `snpeff:`.

> Note: Adjusting presumed upstream parameters e.g., under `align:` will not have the intended effect. Only in special circumstances will the `rules.align.input` be run by itself and only then will you have to adjust parameters in the `align:` section.

The workflow runs on *samples* in *sets* as listed in `samplesets/*.txt` and defined in `/metadata/sample2runlib.csv`. Sample names listed in `samplesets/*.txt` must correspond to the entries in the sample column in `/metadata/sample2runlib.csv`.

6.3.7 *Configuring the Workflow Use Case*

6.3.7.1 *De-novo Analysis*

Option `denovo` can be used to check the relatedness of *sequencing runs* and/or *samples* without the use of any reference genome. It will perform a comparative analysis based on k-

mers on the raw data and output distance matrices. We recommend performing a de-novo analysis at the very start of every project at the "sequencing run"-level prior to any merging of runs into *samples*. This can help to detect mix-ups and mislabels. Run-level clustering is achieved by providing unique names for each sequencing run in the sample column of `metadata/sample2runlib.csv`. De-novo analysis is invoked by calling `rules.denovo.input` (Uncomment the respective line in the `Snakefile`, and only this line). Pay attention to maintain the correct indentation.

```
rule all:
    input:
# USER OPTIONS
        rules.denovo.input,
#         rules.varcall.input,
#         rules.annotate.input,
# EXPERT OPTIONS
#         rules.readqc.input,
#         rules.align.input,
#         rules.stats.input,
```

6.3.7.2 Variant Calling – Standard Re-Sequencing Analysis

Running `rules.varcall.input` will call variants and genotype *samples* with respect to one or several reference genomes. `varcall` will compare *samples* listed in `metadata/samplesets/` with one another, as defined in the sample column. Invoke this analysis by uncommenting `rules.varcall.input` (and only this line). Pay attention to maintain the correct indentation.

```
rule all:
    input:
# USER OPTIONS
#         rules.denovo.input,
        rules.varcall.input,
#         rules.annotate.input,
# EXPERT OPTIONS
#         rules.readqc.input,
#         rules.align.input,
#         rules.stats.input,
```

6.3.7.3 Variant Annotation – The Effects of Variants on Gene Function

If a snpEff library for this (exact) reference genome is provided then the entire workflow can in principle be invoke by uncommenting `rules.annotate.input` (and only this line). Pay attention to maintain the correct indentation.

```
rule all:
    input:
# USER OPTIONS
#         rules.denovo.input,
#         rules.varcall.input,
        rules.annotate.input,
```

```
# EXPERT OPTIONS
#       rules.readqc.input,
#       rules.align.input,
#       rules.stats.input,
```

Currently, `rules.annotate.input` will operate only on one reference genome at a time. The rule will hence propagate upstream only one reference genome as requirement. If variants detected against several different reference genomes need annotating, then first run the `varcall` rule specifying all reference genomes and subsequently invoke the `annotate` rule separately for each reference genome annotation.

6.3.7.4 Expert Options

In cases where only adapter clipping or read alignment is desired, those processes can be invoked separately by the respective rules, `rules.readqc.input` or `rules.align.input`. There are separate configurations sections for those in `config.yml` which must be used. Alignment will trigger prior adaptor clipping. Invoking `rules.stats.input` will produce summary statistics. They will require read alignments and are meaningful for generic `varcall` runs.

6.4 WORKFLOW OUTPUTS

After completion, all output of the workflow, including logs and stats, will be in `output/`.

- Clipped reads (in interleaved fastq) format are in `output/reads/`
- BAM files with In/Del-realigned alignments are in `output/abra/`
- BCF/VCF files of the filtered variants including respective index files are in `output/variants/final/`
- The snpEff-annotated variant file is in `output/annotated_variants/snpeff/`

VCF files in `output/variants/final/` have been filtered with *bcftools view* according to the user defined filter settings in `config.yml`.

For loading into IGV, use the In/Del realigned BAM file in `output/abra/` and the `*.vcf.gz` files of the filtered variants. Note that IGV requires the `vcf.gz.tbi` index.

6.5 SUPPORT

6.5.1 License

The snakemake workflow dna-proto-workflow is released under the MIT License.

6.5.2 Contributors

This workflow was developed by Norman Warthmann, Plant Breeding and Genetics Laboratory of the FAO/IAEA Joint Division (PBGL), with important contributions from Kevin D Murray (Australian National University) and Marcos Conde, PBGL. The documentation was written by Norman Warthmann with contributions from Anibal Morales, PBGL.

6.5.3 Citing

When publishing results obtained using this workflow please cite the link to the original github repository (https://github.com/pbgl/dna-proto-workflow) specifying the release.

6.5.4 Reproducibility

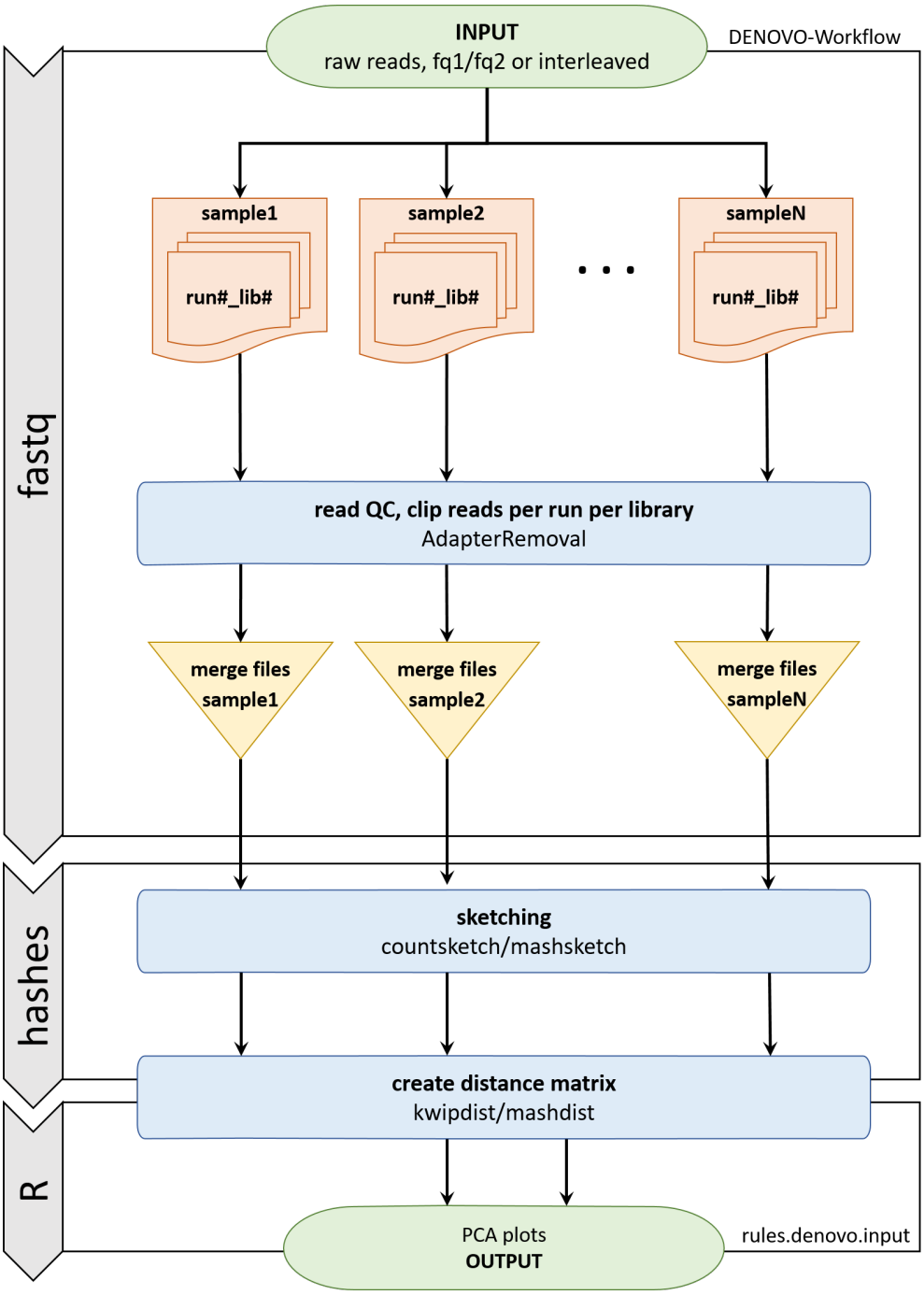
Workflows help addressing reproducibility issues. Consider making your version of the workflow, configured for your data, available upon publication of your results. Check out the archive options of Snakemake.

6.5.5 Getting Help

Feel free to let us know if you are using our workflow and don't hesitate to contact us with questions: email n.warthmann@iaea.org

7 WORKFLOW ILLUSTRATIONS (GRAPHICS)

7.1 DENOVO WORKFLOW



7.2 VARCALL WORKFLOW