Contact prediction workflow using Snakemake

John Lamb *

*Stockholm University, SciLife Lab

Submitted for the Applied Bioinformatics course, as part of the MedBioInfo graduate school

Here we present a snakemake pipeline for automatically making contact predictions from a protein sequence. It comes bundled with jackhmmer as an aligner and ccmpred as a contact predictor. This is fully modularized and can easily be changed to preference. The workflow is set up to take full advantage of the underlying hardware and will automatically use the available cores and run potential steps in parallell to increase speed of execution.

contact prediction | workflow | snakemake

Introduction

Running a contact prediction pipeline from a protein sequence is usually conducted as a multi-step process. Multiple intermittent files are created and running a full dataset of samples requires special attention to create code that make full use of the underlying hardware. To simplify this we have used snakemake to create a scaleable and modular pipeline to run contact predictions from sequence. Snakemake is a workflow management system built to create reproducible and scalable data pipelines. Workflows are described in a human readable, Python based, format.

Materials and Methods

The workflow has been divided up in several modules that each handle one single part, see figure 1. Each sample first go through the alignment-module where the samples are aligned, by default, using **jackhmmer** against a database. The result of this is then fed into the convert-module that converts the output from the aligner to correct format for the chosen predictor. The predictor is by default **ccmpred** and the result from these predictions are fed into the extract-module that extract the top couplings for each sample. All of the samples are then collected into a summary report from which the topcouplings of each sample is accessable.

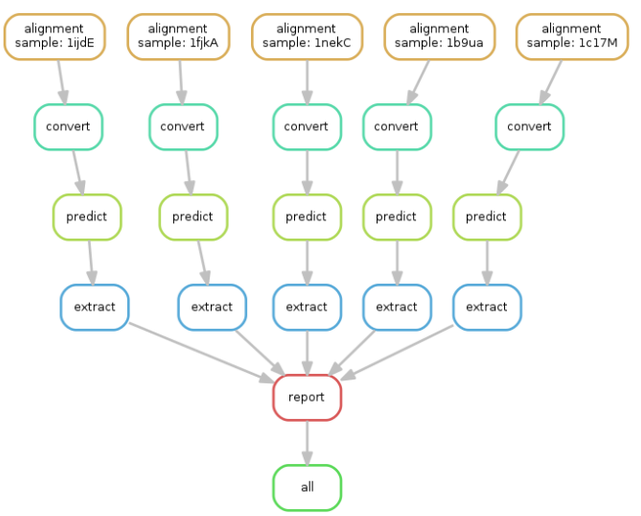

Fig. 1. Workflow schematic with five samples

An important part for scalability is that each module can be specified to use a certain number of cores. Snakemake will automatically try to saturate the available cores so will automatically run modules in parallell if possible. This can clearly be seen when this workflow is run as the alignment-module by default takes the maximum number of cores available and hence the samples are all run in sequence. However, the rest of the modules can only utilise one single core and snakemake will then automatically run these modules in parallell. This does extend to later modules as well so if some samples has finished the convert-module they to not have to wait for the other samples to finish the same module but can progress to the predict-module given there are cores available. This is only true if there are no interdependencies between the samples. The report-module will always wait until all extract-modules has run as it is dependent on all samples having finished.

To make the workflow as modularized as possible each module consists of a separate driver script that runs the underlying code or chosen program. By minimizing the parameters to mostly only input and output directories the choice of program running each module is easily changed. Each driver script is written to return to **stdout** which is then easily captured by snakemake. This means that a change of underlying aligner or predictor is done by a change of call signature in each modules driver script.

All configurations are made in the config.yaml file that specifices input/output directories, database for alignment and any potential limitation on how many cores to use. Snakemake will automatically scale down the number of cores to maximum of available cores. The path to the input directory and the path to the local database file are usually the only two options that needs to be set. The workflow with by default create a results directory in the running directory for all output and the number of cores are set to use all available.

Results

The workflow is fully modularized and deployable and works out of the box with the bundled jackhmmer and ccmpred. By creating a virtual Python3 environment and pip install requirements according to the requirements file and adjusting the database file path to a local database the five test samples run and creates a result folder in the current working directory. This contains all intermediate files together with a summarized report. If these intermediate files are not essential snakemake has an easy feature to stamp intermittend files as temporary and will automatically clean these up. This however will make these files regenerated every time the workflow is run which is not always the wanted result.

Discussion

Snakemake is proven to be a good framework for workflows. It does require some rethinking in the way it functions but as it is similar in principle to GNU make it can quickly be picked up by anyone that has used make before. Even without previous experience it is easy to learn but require a more bottoms-up approach to the workflow than the traditional top-down. A next step in this project is full integration with a running environment and/or container. Snakemake has support for automatic deployment using conda and virtual environment and also to archive fully contained workflows into tarballs for full reproducibility.

As snakemake rules are connected throught the respective input and output files any type of workflow can be generalized. With the use of dummy files any type of directed acyclic workflow could be defined and be run in optimal order and potetial steps be run in parallell automatically handled by snakemakes without the user having to specify anything more than available cores.

1. Köster, Johannes and Rahman, Sven, Snakemake - A scalable bioinformatics workflow engine, Bioinformatics, (2012).
2. R.D. Finn et al., HMMER Web Server: 2015 update, Nucleic Acids Research, Web Server Issue 43 (2015), W30–W38.
3. Seemayer S, Gruber M, Söding J., CCMpred – fast and precise prediction of protein residue-residue contacts from correlated mutations, Bioinformatics, 30 (2014), pp. 3128–3130
4. Skwark MJ, Raimond D, Michel M, Elofsson A, Improved Contact Predictions Using the Recognition of Protein Like Contact Patterns, PLOS Comput Biol, (2014), e1003889
5. Michel M, Menéndez Hurtado D, Uziela K, Elofsson A, Large-scale structure prediction by improved contact predictions and model quality assessment., Bioinformatics, 33 (2017), pp. 23–29