snakemake

A framework for reproducible data analysis

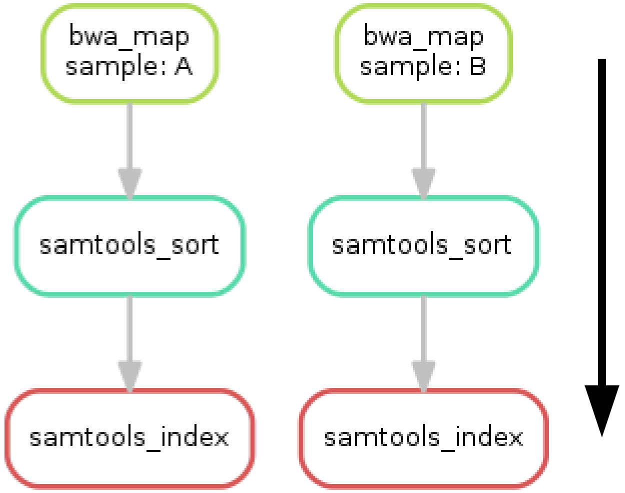
Why use it?

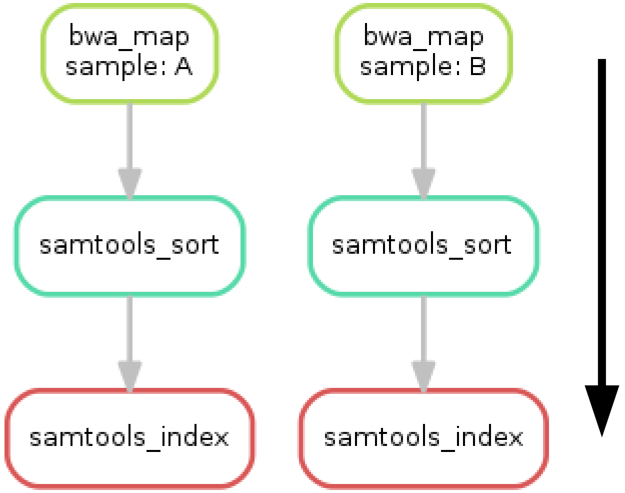
- ***Reproducibility*** – Snakefiles can be run in self-contained Conda (Mamba) or Singularity enviornments and all parameters are fixed, can record logs and version control etc.
- ***Scalibility*** **–** Given the right config files, Snakemake can automatically request optimal resources to run jobs on a given platform, without the need to modify the workflow.
- ***Interpretability*** – Built using extended Pythonic language, so can run standard Python code alongside Snakemake rules and thus leans on all the benefits of Python.

- Can pause and run jobs in the middle of processing (in theory, see later)

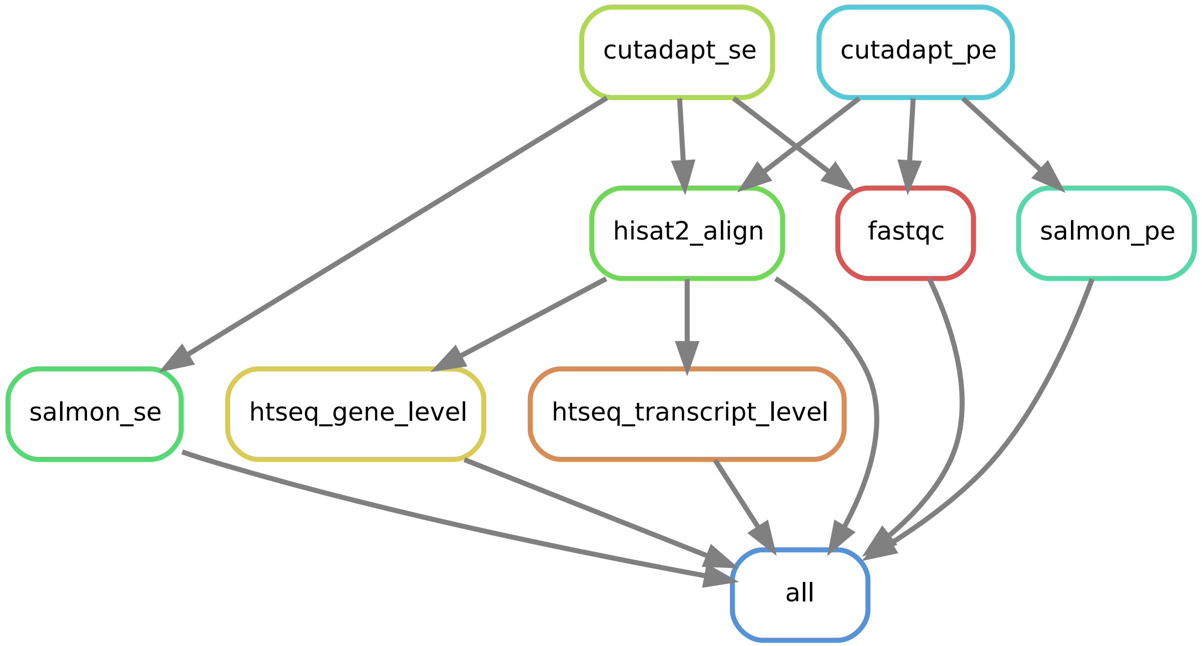
How Does it Work?

- Builds on the concept of GNU Make, for installing packages via the Command-Line. Workflows are defined as rules that define how to create output files from input files.
- Dependencies determined automatically via matching filenames, creating a DAG (Directed, Acyclic Graph) of jobs that can be parallelised.
- Filenames are matched between rules via the use of a dictionary-like object of “wildcards”, or `{wildcards}`, which contain strings that can be added to to define eg .fastq.gz files, .bam files, .csv files etc
- Rules contain functions/commands (ie Cutadapt QC reads, HISAT2 align reads etc) written in the underlying Shell script (ie Bash for Linux).

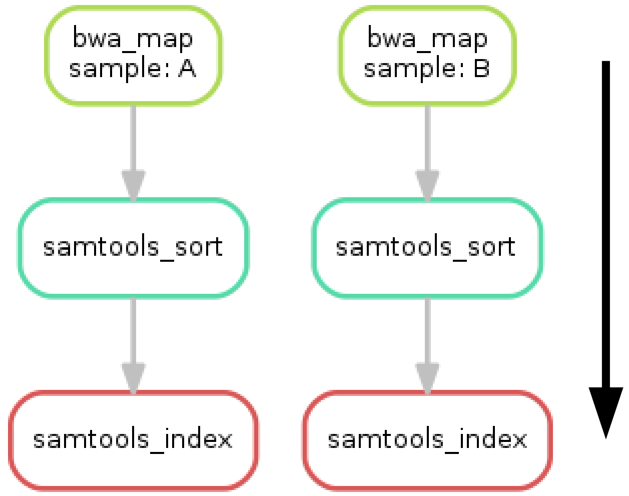

bwa_map
sample: A
bwa_map
sample: B
samtools_sort
samtools_sort
samtools_index
samtools_index

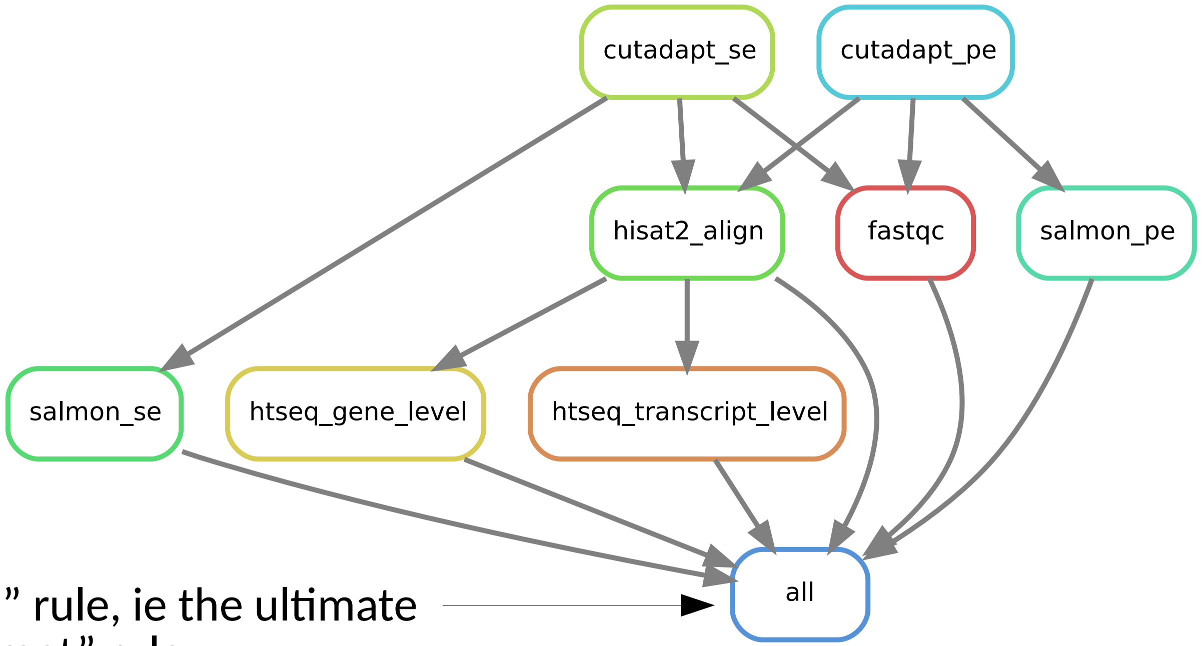


```
snakemake --forceall --rulegraph | dot -Tsvg > dag.svg
```



```
snakemake --forceall --rulegraph | dot -Tsvg > dag.svg
```





"All" rule, ie the ultimate
"Target" rule

Look at Snakefile

How to Use it?

- Recommend using Atom editor, as it recognises Snakefiles
- Can run with the `-n` flag, which runs Snakemake via a “dry run”, where shell is not run, no files are generated and no jobs submitted, just tests the syntax and layout of the Snakefile
- Will need a specific script to run on the HPC (ARC4), a “cluster config” fil
- `--cores 80` means max 80 cores allocated, split 4 per task (so 20 at once). Will optimise if spare threads. Can use `--resources` to allocate other things ie memory/GPU usage etc
- Can overwrite a lot of arguments in the rules from command line, ie definite input files or set threads with `--set-threads my_rule=2`

```
snakemake --profile sge --cluster-config yamls/cluster.yaml --use-conda --rerun-incomplete --cores 80
```

Tips

- Config files! YAML or JSON, recommend reading into them
- Can use specific Conda yamls, or can just use same Conda env with all require packages & Snakemake intstalled
- I generally have “snakemake” Conda env containing just Snakemake, run from there and using .yamls specifying Conda envs for different rules
- Mamba! An optimised form of Conda, basically works the same but is multi-threaded and speedier (in theory)
- `snakemake --lint` will automatically attempt to indicate where the Snakefile can be optimised

- Tutorial online is really helpful and comprehensive! https://snakemake-wrappers.readthedocs.io/en/stable/index.html
- Multiple published pipelines already exist online, especially for more routine things like RNA-Seq – don’t need to re-invent the wheel!

Problems?

- Can be very fiddly! Making sure all filenames match etc, and error messages can be really unhelpful and obscure
- While `-n` option lets you check if the Python/Snakemake script works, doesn't run or account for shell scripts, which has to be done manually via checking output logs
- Requires an open internet connection to HPC to run; obviously challenging given WFH and general instability of connection. If the connection is dropped then the pipeline totally fails so have to totally re-run, hence my usage of `--rerun-incomplete`

Thanks!

Any Questions?

Euan McDonnell
bs14e3m@leeds.ac.uk
@EuancRNA

Thanks to Joseph Barker for setting up Snakemake on the HPC and writing the cluster.yml config file