Setup

Create environment and activate it

conda activate base

mamba create -c conda-forge -c bioconda -n name snakemake

conda activate name

Rule syntax

rule rule.name:

input:

“input.one”

“input.two”

output:

“output.one”

“output.two”

params:

#set parameters here (for example, “pval=0.05”)

parameter=value

run:

#what to run

shell:

#python string containing shell commands to execute

#here you can refer to inputs or outputs put putting them in curly brackets

#example: “bwa mem {input} | samtools view -Sb - > {output}”

#when multiple inputs, they are concatenated with white space in between

script:

#path and filename to custom script

#python scripts: refer to inputs as: snakemake.input[num]

#remember that python starts with 0 as the first one

#r scripts: call with snakemake@input[[num]]

#starts with one here, because it’s in R

#or do snakemake@input[["filename"]]

Commands

conda activate snakemake #enter snakemake environment

Ctrl+C #exit running snakemake workflow (kills all runs)

conda deactivate #deactivate active snakemake environment

snakemake --help #see available tools

snakemake directory/file

#tells snakemake to generate the target files (this can be a rule output)

#options

-n #dry run (show execution plan without actually running)

-p #print the resulting shell command

--cores number #specify number of cores to use

--dag #visualize the DAG

#pipe into dot -Tsvg > dag.svg to save the dag visualization

expand(“directory/{something}.bam”, something=NAME) #helper funct for collecting input files

Best practice

#at top of file

rule all:

input:

"directory/finaltargetfile"

snakemake --lint #invokve the code quality checker (linter)

#apply the formatter Snakefmt before publishing

#when publishing to github. Add some minimal test data and configure github actions

#stick to a standardized structure when publishing a workflow

#configuration handled via config files

#keep filenames short but informative

#stick to a single special character as a separator throughout the whole thing (\_ or -)

#keep python code like helper functions separate from rules

#use wrappers when possible

#recommended to store each workflow in a dedicated git repo with this structure:

├── .gitignore

├── README.md

├── LICENSE.md

├── workflow

│ ├── rules

| │ ├── module1.smk

| │ └── module2.smk

│ ├── envs

| │ ├── tool1.yaml

| │ └── tool2.yaml

│ ├── scripts

| │ ├── script1.py

| │ └── script2.R

│ ├── notebooks

| │ ├── notebook1.py.ipynb

| │ └── notebook2.r.ipynb

│ ├── report

| │ ├── plot1.rst

| │ └── plot2.rst

| └── Snakefile

├── config

│ ├── config.yaml

│ └── some-sheet.tsv

├── results

└── resources

Config

configfile: ‘config.yaml’

#must create config.yaml file in the directory

#creates an object called “config”, within which values are accessible

#seems to read in as dictionary

#can even have stuff in the config file that are lists or dictionaries

#example config file

name: value

dictionary\_name:

Key1: “value1”

Key2: “value2”

#another example

Thresholds:

Pvalue: 0.05