Snakemake

# creating a snakefile

A snakefile can be created at any location, just create a file called “Snakefile” without an extension.

# SNakefile structure

A Snakefile is collection of rules (steps to preform during the analysis) and is written in a combination of python and bash, meaning that raw python scripts can be included at the start or end of a Snakefile (figure 1). Bash-scripting can be used inside the rules specified in the snakefile.

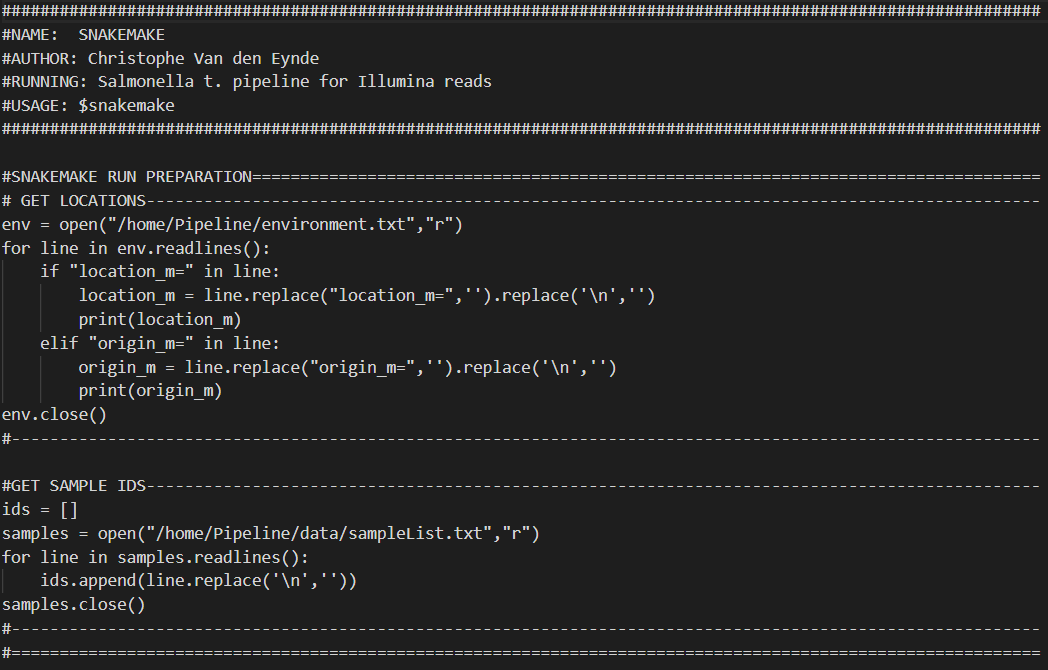
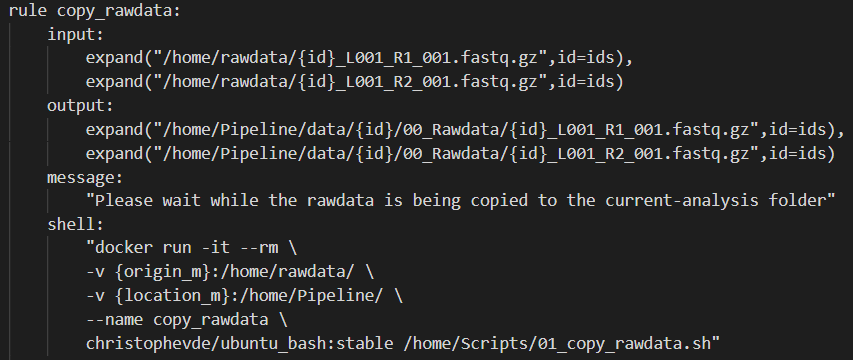


Figure 1: some python scripting at the start of the snakefile

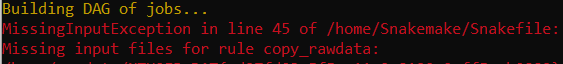
## rules



### rule chain

A snakefile rule-chain starts with a “rule all” which is a master rule, the input files/folders of this rule will always be the output files/folders of other rules. This rule is the final rule to be executed in the pipeline, but it’s the rule that will call all other rules in the snakefile in order to get the required input.   
Snakemake follows the pull-strategy where each rule will search and activate the rule required to get the requested input. This means that the rules themselves don’t really need to be in any specific order inside the snakefile. The only thing to think about is that the first rule specified in the snakefile will always be seen as the master rule, the “rule all”.

### basic rule structure:

1. Each **rule** will start by specifying its **name**, these names need to be unique
2. **Input**:  
   Next the **Input** will be specified, these are some files/folders that the rule needs in order to be activated (most of the time the input will be the same as the output of another rule). If these files aren’t available, the rule will either execute the rule(s) that will provide the required files or throw an error:  
   
3. **Output**:  
   Specifying the **output** is needed to chain multiple rules together and is done in the same way as the input is specified. If the output files aren’t found, Snakemake will throw an error and delete all generated output files.
4. **Message**:  
   The **message** tag is optional. Under this tag some one can provide some text that will be echoed/printed in the terminal
5. **Shell**:  
   Under the shell tag, different shell commands can be specified that need to be executed in order to get the output. One could even create an entire bash script under this tag if desired.

Almost everything specified under a tag needs to be between “ ”, however there are some exceptions (mostly when using wildcards).

### Wildcards and directory’s

Wildcards can be used in a snakefile and will mostly be the result of raw python code somewhere in the snakefile.

* **Normal** wildcards will mostly be specified with {*name*}.
* If the wildcard is a **list**, or another itterable object, the expand function is required. The **expand** function will iterate over all values in the list. The wildcard itself will still be specified with {*element*} within the string, but after the string a “*,element=list*” is required, specifying which list to iterate over.

Directory’s need to be specified using the directory tag: *directory(“location”).* Directory’s can only be specified in the output and can’t be used as input. If a directory isn’t specified like this or when a file is specified as a directory, docker will throw an error: “unexpected file type”.

Wildcards and directory specification can be chained together like this, notice that neither the directory() or the expand() are between “”:  
Directory(expand(“location/{folder}”,folder=folders))