Reproducible data analysis with Snakemake

Johannes Köster
2019

https://koesterlab.github.io
150k downloads since 2015
497 citations (+150 in 2019)

https://snakemake.readthedocs.io
rule mytask:
  input:
    "data/{sample}.txt"
  output:
    "result/{sample}.txt"
  shell:
    "some-tool {input} > {output}"
rule mytask:
  input:
    "data/{sample}.txt"
  output:
    "result/{sample}.txt"
  script:
    "scripts/mytask.R"
rule mytask:
  input:
    "data/{sample}.txt"
  output:
    "result/{sample}.txt"
  script:
    "scripts/mytask.jl"
import matplotlib.pyplot as plt
import pandas as pd
d = pd.read_table(snakemake.input[0])
d.hist(bins=snakemake.config["hist-bins"]) plt.savefig(snakemake.output[0])

rule mytask:
    input:
        "data/{sample}.txt"
    output:
        "result/{sample}.txt"
    script:
        "scripts/mytask.py"
rule mytask:
    input:
        "data/{sample}.txt"
    output:
        "result/{sample}.txt"
    script:
        "scripts/mytask.py"

import matplotlib.pyplot as plt
import pandas as pd

d = pd.read_table(snakemake.input[0])
d.hist(bins=snakemake.config["hist-bins"])  
plt.savefig(snakemake.output[0])

https://snakemake.readthedocs.io
rule mytask:
    input: "path/to/{dataset}.txt"
    output: "result/{dataset}.txt"
    script: "scripts/myscript.R"

rule myfiltration:
    input: "result/{dataset}.txt"
    output: "result/{dataset}.filtered.txt"
    shell: "mycommand {input} > {output}"

rule aggregate:
    input:
        "results/dataset1.filtered.txt",
        "results/dataset2.filtered.txt"
    output: "plots/myplot.pdf"
    script: "scripts/myplot.R"
rule mytask:
    input:
        "path/to/{dataset}.txt"
    output:
        "result/{dataset}.txt"
    conda:
        "envs/mycommand.yaml"
    shell:
        "mycommand {input} > {output}"

channels:
    - bioconda
    - conda-forge

dependencies:
    - mycommand = 2.3.1

https://snakemake.readthedocs.io
rule mytask:
    input:
        "path/to/{dataset}.txt"
    output:
        "result/{dataset}.txt"
    singularity:
        "docker://some/container"
    shell:
        "mycommand {input} > {output}"
rule mytask:
  input:
    "path/to/{dataset}.txt"
  output:
    "result/{dataset}.txt"
  conda:
    "envs/mycommand.yaml"
  singularity:
    "docker://some/os"
  shell:
    "mycommand {input} > {output}"
## Allele Frequency Estimation

### Results

<table>
<thead>
<tr>
<th>File</th>
<th>Size</th>
<th>Description</th>
<th>Job properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>simulated-bwa.DEL.svg</td>
<td>328.3 KB</td>
<td>Allele frequency estimation error vs. true allele frequency on simulated data.</td>
<td>plot_allelefreq</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>run=simulated-bwa,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>vartype=DEL</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>varlociraptor_callers=[</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>len_ranges=[[1, 30],</td>
</tr>
<tr>
<td>simulated-bwa.DEL.svg</td>
<td>1.9 MB</td>
<td>Allele frequency estimation error vs. true allele frequency on simulated data. The dashed lines depict the standard deviation, solid line depicts the mean. The grey area shows the standard deviation.</td>
<td>plot_allelefreq_scatter</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>run=simulated-bwa,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>vartype=DEL</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>depth_ranges=[[1, 20], [20, 40]], callers=[delly, 'lancet', 'manta', 'streika']</td>
</tr>
</tbody>
</table>
# Snakemake Report

## Workflow

<table>
<thead>
<tr>
<th>Rule</th>
<th>Jobs</th>
<th>Output</th>
<th>Singularity</th>
<th>Conda environment</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>adhoc_varlociraptor</td>
<td>40</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>aggregate_concordance</td>
<td>30</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```python
from common import load_variants
import networkx as nx
import pandas as pd
import numpy as np

type = snakemake.wildcards.type
index_cols = ['CHROM', 'POS', 'SVLEN'] if var
all_variants = [load_variants(f, vartype=var)
G = nx.Graph()
for calls, (i, j) in zip(all_variants, snakemake)
calls['component'] = None
for call in calls.iteritems():
a = i, call, Index
g.add_node(a)
if call.MATING == 0:
b = j, call.MATING
G.add_node(b)
g.add_edge(a, b)

# get a set of calls for each dataset (we do
representatives = {snakemake.params.dataset_c
if snakemake.wildcards.mode != 'varlociraptor
varlociraptor_variants = [load_variants(f
for calls in varlociraptor_variants:
calls.set_index(index_cols, inplaces=True)
varlociraptor_repre
```

https://snakemake.readthedocs.io