Reproducible data analysis with Snakemake

Johannes Köster
2019

https://koesterlab.github.io
Snakemake is popular

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.py"
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"
```python
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])
```
input: mytask:
output: "data/{sample}.txt"
script: "result/{sample}.txt"

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:
    "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"
Scalability

workstation

compute server

cluster

gird computing

cloud computing

https://snakemake.readthedocs.io
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

**Show: 10 entries**

<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>Rule: plot_allelefreq</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Wildcards: 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>Params: varlociraptor_callers=[</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>'delly', 'lancet', 'manta', 'streika'],</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>len_ranges=[[1, 30], [30, 50],</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>[50, 100], [100, 250]]</td>
</tr>
<tr>
<td>simuulated-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>Rule: plot_allelefreq Scatter</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Wildcards: 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>Params: depth_ranges=[[1, 20],</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>[20, 40]], callers=[ 'delly',</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>'lancet', 'manta', 'streika']</td>
</tr>
</tbody>
</table>

https://snakemake.readthedocs.io
### Snakemake Report

<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
vartype = snakemake.wildcards.vartype
index_cols = ["CHROM", "POS", "SVLEN"]
if var
    all_variants = [load_variants(f, vartype=vartype)
        G = nx.Graph()
        for calls, (i, j) in zip(all_variants, snakemake.calls["component"] = None
            for call in calls.itertuples():
                a = (i, call.Index)
                G.add_node(a)
                if call.MATCHING >= 0:
                    b = (j, call.MATCHING)
                    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()]
            for calls in varlociraptor_variants:
                calls.set_index(index_cols, inplace=True)
        varlociraptor_representatives = {snakemake
```
portability

automation/documenation

results

dataset

dataset

dataset

dataset

dataset

https://snakemake.readthedocs.io