Genomics and Snakemake

FAES BIOF309: Introduction to Python Programming

April 23, 2019

Bari J. Ballew, PhD

- **What is genomics?**
- How do you run a reproducible, automated workflow?
- What is snakemake?
- Interactive demo

What is genomics?

- Genomics is the study of genomes - structure, function, evolution, etc.
- What can genomics do?

FDA Approves Merck's KEYTRUDA® (pembrolizumab) in ...
Associated Press - Apr 22, 2019
Treatment with the **KEYTRUDA**-axitinib combination continued until .... Patients with EGFR or ALK **genomic** tumor aberrations should have ...

Bereaved parents find joy with new baby, thanks to NIH
The Times of Israel - Jun 20, 2014
NEW YORK (JTA) – Even before their daughter, **Ayelet Galena**, was diagnosed with a rare bone marrow disease called dyskeratosis congenita

Genome Editing and World Hunger
Technology Networks - Mar 31, 2019
Genome editing techniques in particular, such as CRISPR/Cas, could help to make **agriculture** more productive and environmentally friendly.

The ingenious and 'dystopian' DNA technique police used to hun...
Washington Post - Apr 27, 2018
For decades, **police** say, the **DNA** of the "Golden State Killer" sat in evidence storage — a unique genetic fingerprint that could identify ...

- Even more cool stuff: https://www.genome.gov/dna-day/15-for-15

Whole genome sequencing has become ubiquitous

- Human Genome Project: “finished” in 2003
  - Approx. 4 years and $500 million[1]

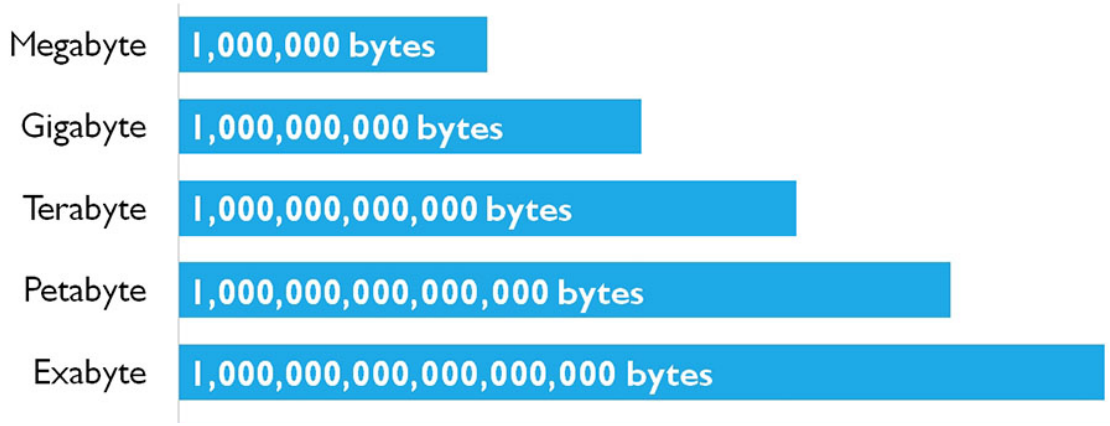
- Whole genome sequencing today (2019):
  - A few days and $1000

- BUT…these are just the costs to go from cell to raw sequence data
  - It’s like when you’ve created some cell lines: that’s not the end – it’s the beginning!

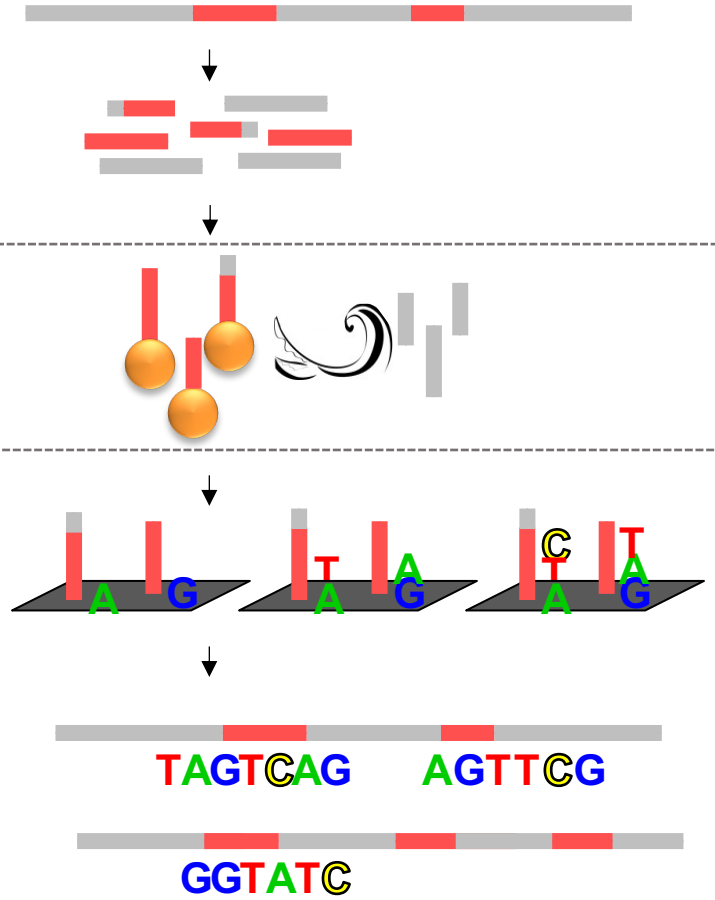
[1] https://www.genome.gov/27565109/the-cost-of-sequencing-a-human-genome/

Additional costs of whole genome sequencing

- Data storage and access
  - One patient's tumor and normal sample = 1 TB of data
- Processing power
  - We have tools that can take advantage of powerful parallel environments, but it costs money to have access to such environments
- Analysis
  - Bioinformaticians to run, develop, maintain tools and pipelines
  - So you have data…what does it mean? (Job security!)
    - Burden testing, clustering analysis, mutational signatures, tumor driver analysis, pathway analysis, etc etc…



Basics of DNA sequencing



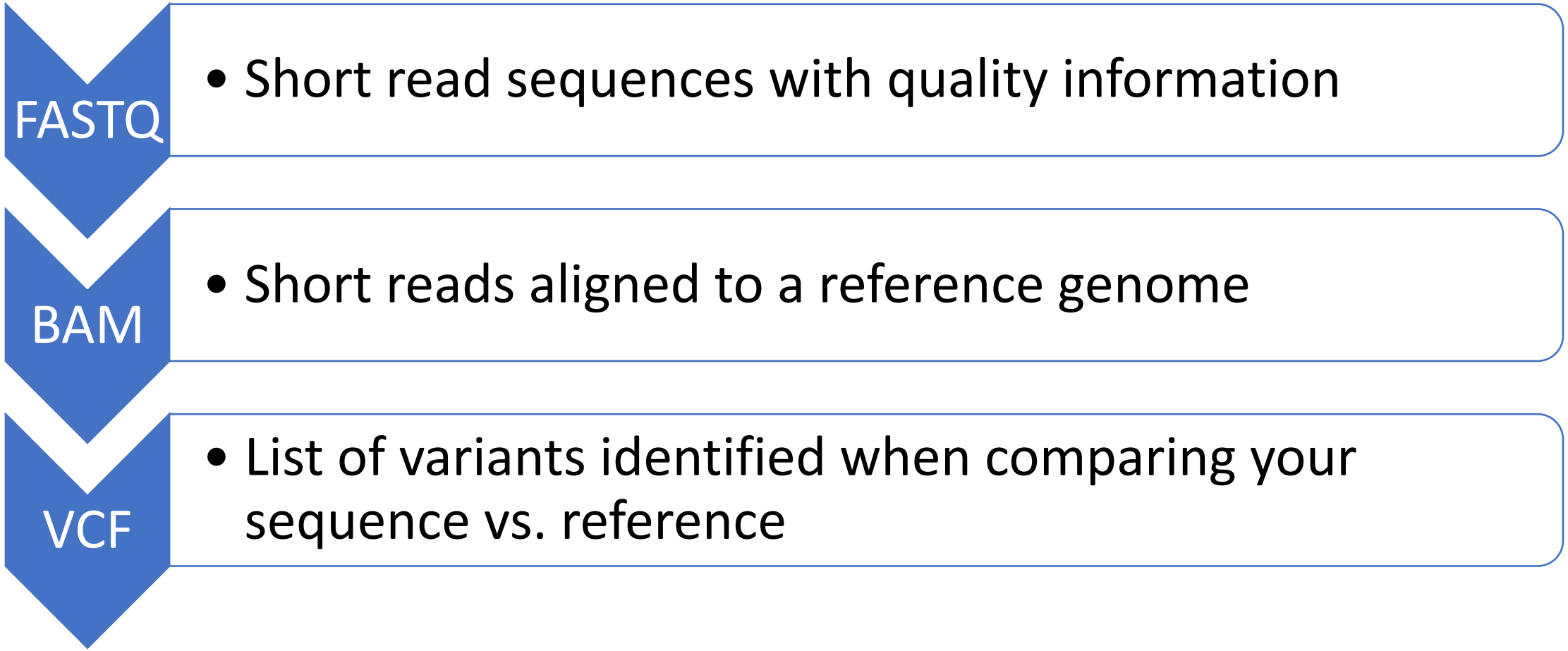
A patient's genomic DNA, composed of introns and exons, is fragmented

FOR EXOME ONLY: A commercial kit is used to create an exon-enriched library

That library is amplified and then sequenced

Reads are aligned to the reference genome; software looks for discrepancies between the reads and the reference sequence (SNPs – single nucleotide polymorphisms)

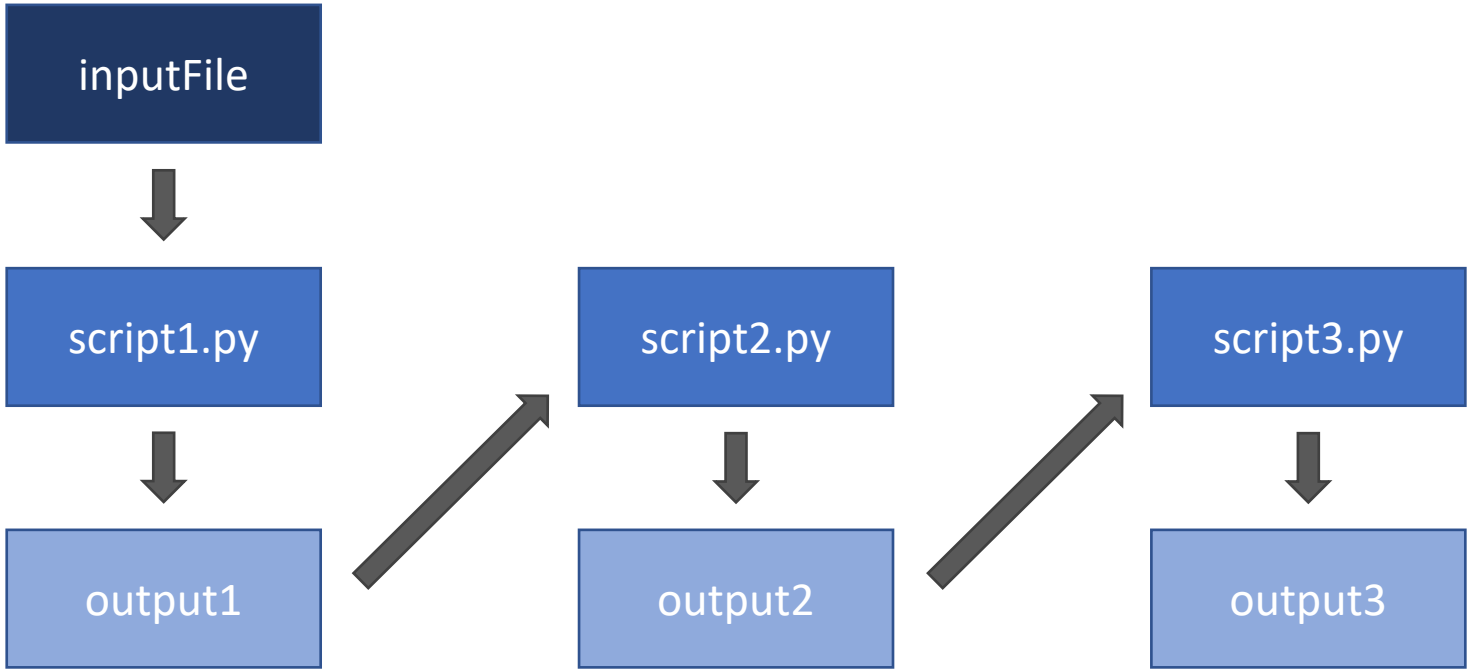
Basics of DNA sequencing



- What is genomics?
- **How do you run a reproducible, automated workflow?**
- What is snakemake?
- Interactive demo

Without a pipeline

- Let’s say you have three python scripts, that need to be run in order, to produce some data:



Without a pipeline

- Let's run it!

```
ballewbj@cgemsIII $ python /path/to/script1.py /path/to/inputFile /path/to/output1
ballewbj@cgemsIII $ python /path/to/script2.py /path/to/output1 /path/to/output2
ballewbj@cgemsIII $ python /path/to/script3.py /path/to/output2 /path/to/output3
```

- And now you have to do it again:

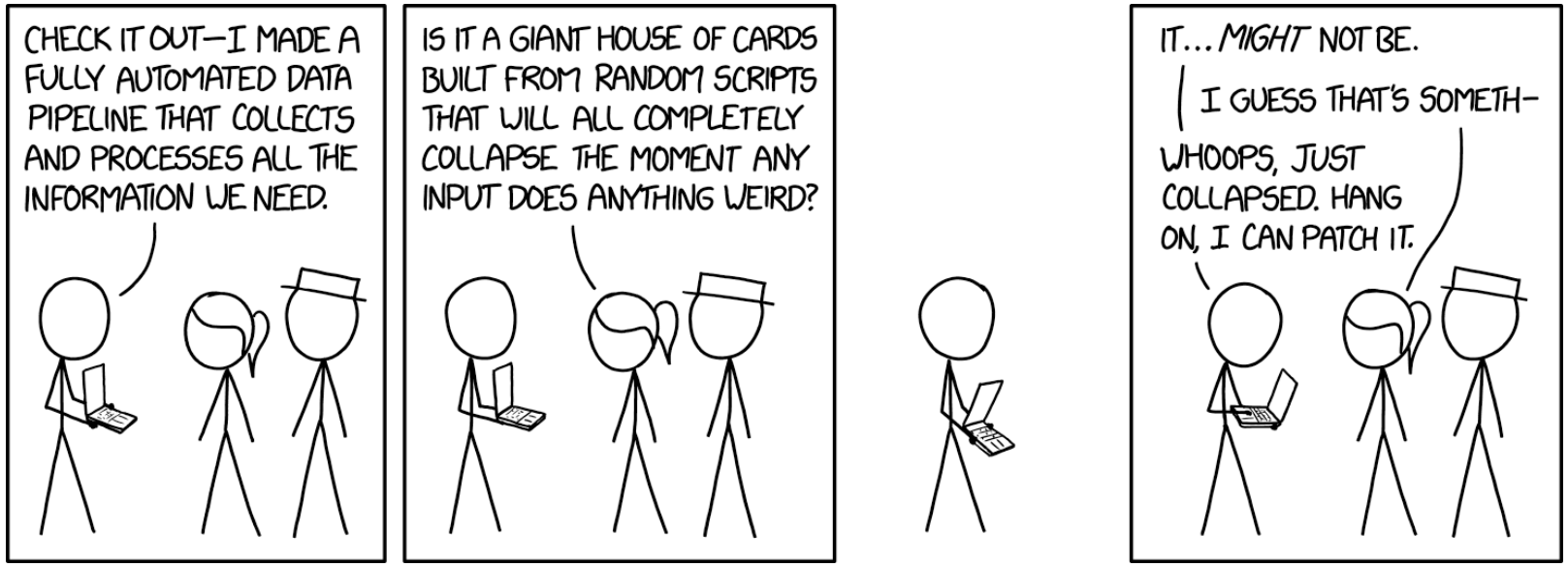
```
ballewbj@cgemsIII $ python /another/path/to/script1.py /another/path/to/inputFile /another/path/to/output1
ballewbj@cgemsIII $ python /another/path/to/script2.py /another/path/to/output1 /another/path/to/output2
ballewbj@cgemsIII $ python /another/path/to/script3.py /another/path/to/output2 /another/path/to/output3
```

- And again:

```
ballewbj@cgemsIII $ python /path/to/script1.py /path/to/inputFileA /path/to/outputA
ballewbj@cgemsIII $ python /path/to/script2.py /path/to/outputA /path/to/outputB
ballewbj@cgemsIII $ python /path/to/script3.py /path/to/outputB /path/to/outputC
```

- Now, do it 300 more times. How many typos will you make?

Pipeline: reproducible, automated workflow



https://xkcd.com/2054/

Writing a pipeline from scratch

- Let's create a rudimentary pipeline for our task as a bash script called pipeline.sh:

```
inPath=$1
outPath=$2
sampleName=$3

python /path/to/script1.py ${inPath}/${sampleName} ${outPath}/${sampleName}_out1
python /path/to/script2.py ${outPath}/${sampleName}_out1 ${outPath}/${sampleName}_out2
python /path/to/script3.py ${outPath}/${sampleName}_out2 ${outPath}/${sampleName}_out3
```

- You can run it via

```
ballewbj@cgemsIII $ bash pipeline.sh /my/in/path /my/out/path sampleA
```

Writing a pipeline from scratch

We can run this shell script and all three steps of our pipeline will be run. But to avoid creating a "house of cards that will collapse the moment any input does anything weird," we also have to write checks for:

- Did we provide some value for inPath, outPath, and sampleName? Are there any weird characters in the values, like spaces in the path?
- Does the input file exist, and can we read it?
- Do inPath and outPath exist? If they do, can we write to them? If they don't, can we make them? If we make them, was that successful?
- Did script1.py run successfully? What about script2.py or script3.py? If script1.py fails, script2.py and script3.py will try to run anyway. What errors will they generate? How will we know what happened?
- If we try to run pipeline.sh and an output file already exists, how will we handle that?
- If we run this on 500 files and some complete successfully but some don't, can we resume the pipeline and only re-run the samples that failed? How will we detect failed vs. successful jobs?
- What if we have lots of steps, and some can happen concurrently, but others are dependent on preceding steps?
- If our scripts start up sub-routines, the original script may finish executing but the sub-routines are still running to create the output file. How will our pipeline know that the results are ready and the next script may start?

Writing a pipeline from scratch

DON’T!

(unless you really really really know what you are doing)

Workflow management systems

- Create **reproducible**, **scalable**, and **portable** pipelines
  - **Reproducible**: explicit documentation of tools, versions, and parameters used in analysis
  - **Scalable**: handle parallelization and multi-threading
  - **Portable**: can run locally, on HPC, on cloud without changing the workflow
- Execute analysis without manual intervention
- Have built-in infrastructure to handle most of the questions described in our previous example
- Many examples (CWL, WDL, NextFlow, snakemake, make)

- What is genomics?
- How do you run a reproducible, automated workflow?
- **What is snakemake?**
- Interactive demo

Snakemake

- Pythonic workflow management system
- Scalable, portable, reproducible
- Built-in dependency management tools (conda, containers)

Snakemake basics: rules

```
inPath = '/path/to/input/'
outPath = '/path/to/output/'
sampleName = 'patient1'

rule run_script1:
    input:
        inPath + sampleName
    output:
        outPath + sampleName + '_out1'
    shell:
        'python3 script1.py {input} {output}'
```

- Each step of your analysis becomes a “rule” with defined input and output
- A rule will not run until the input files are present
- Rules are not considered complete until the output files are present and the shell command exits without error

Snakemake basics: rules

```
inPath = '/path/to/input/'
outPath = '/path/to/output/'
sampleName = 'patient1'

rule run_script1:
    input:
        inPath + sampleName
    output:
        outPath + sampleName + '_out1'
    params:
        num = '3'
    shell:
        'python3 script1.py -n {params.num} {input} {output}'
```

- Rules can have additional/alternative components, like:
  - **params**: to define any other information that needs to be passed to the command being run

Snakemake basics: rules

```
inPath = '/path/to/input/'
outPath = '/path/to/output/'
sampleName = 'patient1'

rule run_script1:
    input:
        inPath + sampleName
    output:
        outPath + sampleName + '_out1'
    threads: 8
    shell:
        'python3 script1.py -t {threads} {input} {output}'
```

- Rules can have additional/alternative components, like:
  - **params**: to define any other information that needs to be passed to the command being run
  - **threads**: for multi-threading tasks

Snakemake basics: rules

```
inPath = '/path/to/input/'
outPath = '/path/to/output/'
sampleName = 'patient1'

rule run_script1:
    input:
        inPath + sampleName
    output:
        outPath + sampleName + '_out1'
    run:
        with open(str(input), 'r') as i, open(str(output), 'w') as o:
            for line in i:
                line = line[:-1]
                o.write('\t'.join([line, 'copied']) + '\n')
```

- Rules can have additional/alternative components, like:
  - **params**: to define any other information that needs to be passed to the command being run
  - **threads**: for multi-threading tasks
  - **run** instead of **shell**: to run native python

Snakemake basics: flow of logic

```
inPath = '/path/to/input/'
outPath = '/path/to/output/'
sampleName = 'patient1'

rule all:
    input:
        outPath + sampleName + '_out2'

rule run_script1:
    input:
        inPath + sampleName
    output:
        outPath + sampleName + '_out1'
    shell:
        'python3 script1.py {input} {output}'

rule run_script2:
    input:
        outPath + sampleName + '_out1'
    output:
        outPath + sampleName + '_out2'
    shell:
        'python3 script2.py {input} {output}'
```

- Snakemake works backwards from a target file, specified in "**rule all**"
- If snakemake can find a way to generate the target file, it will run the workflow
  - The output of **rule run_script1** is the input of **rule run_script2**
  - The output of **rule run_script2** is the input for **rule all** (aka the target file)

Snakemake basics: multiple samples

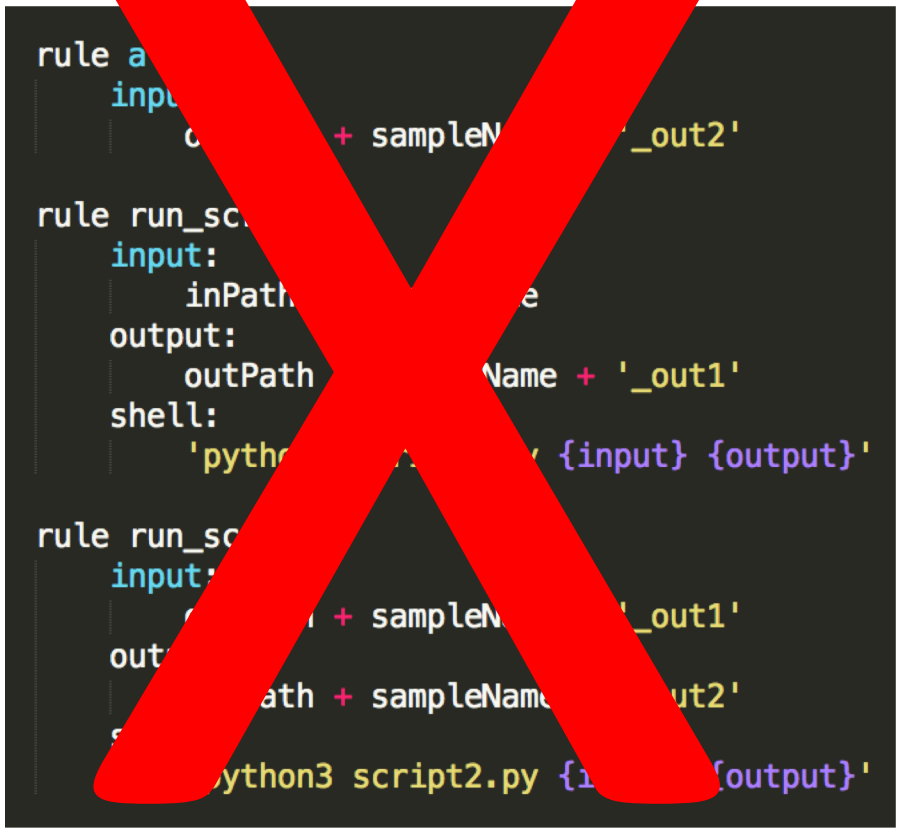
```
inPath = '/path/to/input/'
outPath = '/path/to/output/'
sampleList = ['patient1', 'patient2', 'patient3']
```

- What if we want to apply our rules to more than one sample?

Snakemake basics: multiple samples

```
inPath = '/path/to/input/'
outPath = '/path/to/output/'
sampleList = ['patient1', 'patient2', 'patient3']
```

```
for sam  ame in sample :
```



- What if we want to apply our rules to more than one sample?

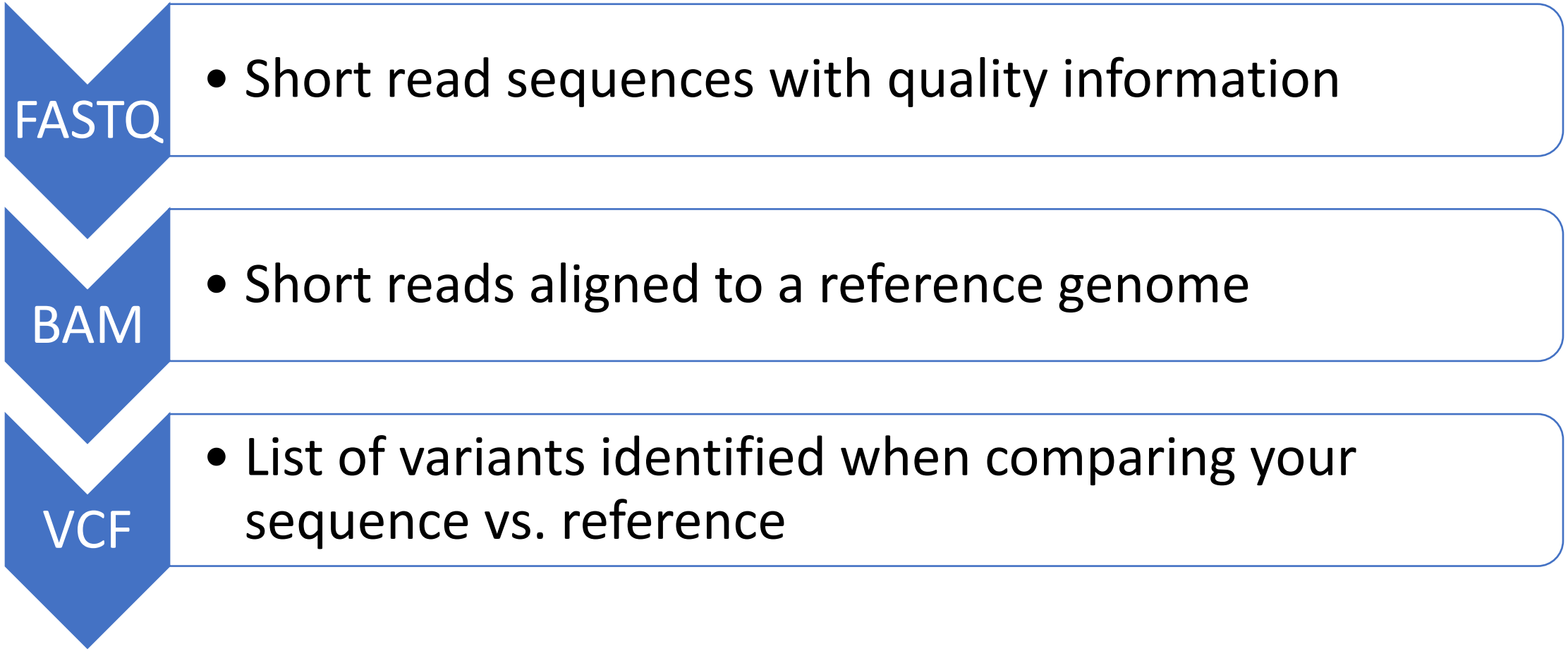
Snakemake basics: multiple samples

```
inPath = '/path/to/input/'
outPath = '/path/to/output/'
sampleList = ['patient1', 'patient2', 'patient3']

rule all:
    input:
        expand(outPath + '{sample}_out2', sample=sampleList)

rule run_script1:
    input:
        inPath + '{sample}'
    output:
        outPath + '{sample}_out1'
    shell:
        'python3 script1.py {input} {output}'

rule run_script2:
    input:
        outPath + '{sample}_out1'
    output:
        outPath + '{sample}_out2'
    shell:
        'python3 script2.py {input} {output}'
```

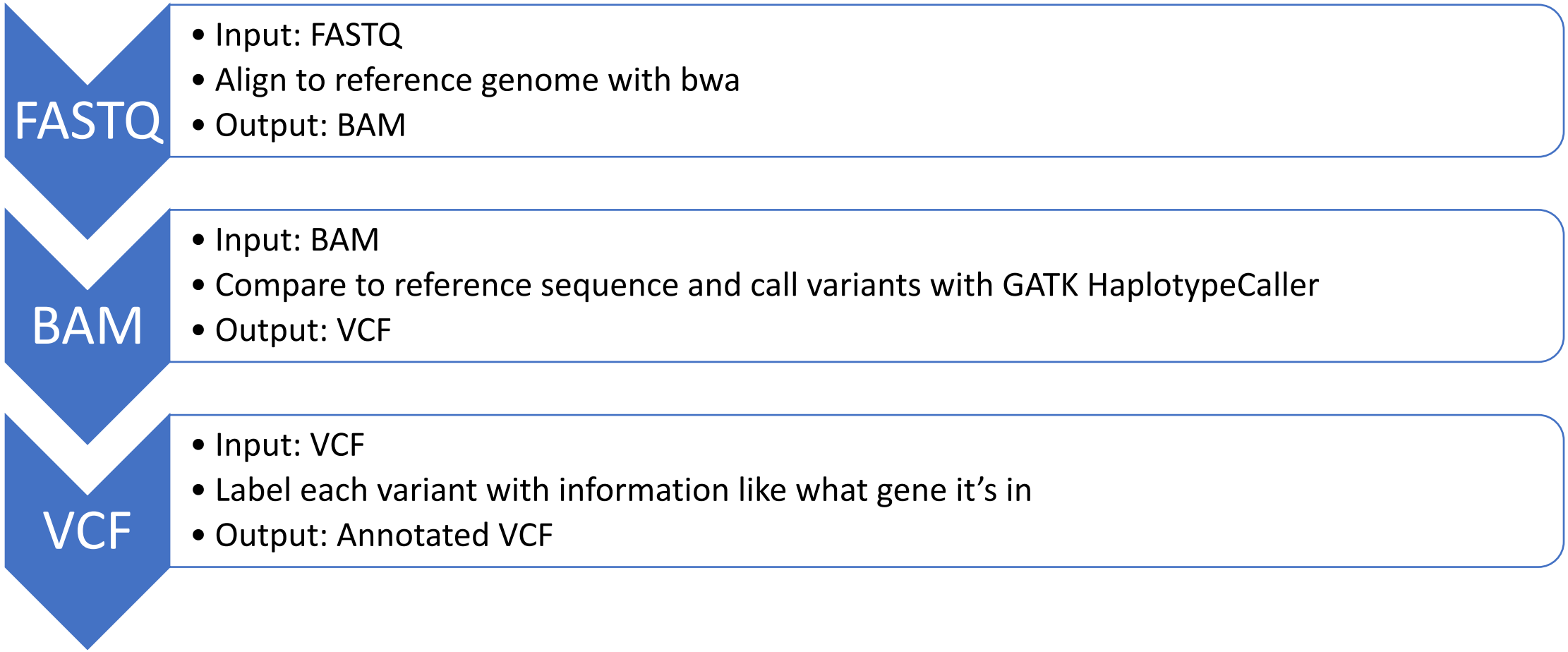
- There's no need to write loops for snakemake to run rules on multiple samples!
- Generalize rules using **wildcards** (in this case '{sample}')
- A wildcard gets defined by the last rule it's used in, using an **expand** statement

- What is genomics?
- How do you run a reproducible, automated workflow?
- What is snakemake?
- **Interactive demo**

Basics of DNA sequencing refresher



Simple DNA-seq pipeline



Options for the demo (pick one!):

- Go to https://github.com/marskar/snakemake/tree/master and click on the JupyterLab binder button

JupyterLab: launch binder

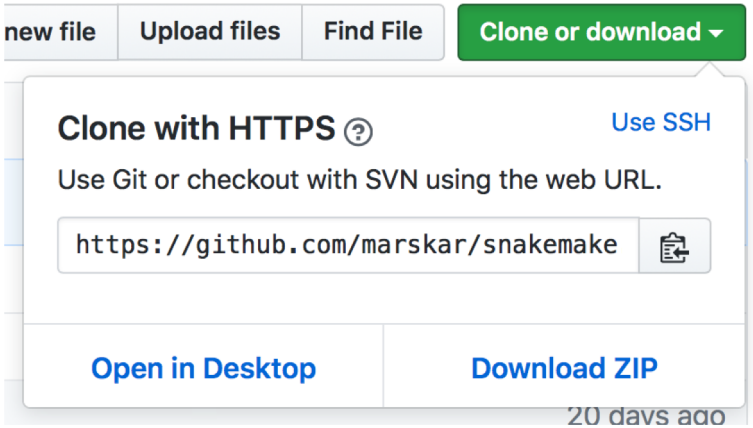
- Run in a conda environment on your local machine
  1. Clone the repo linked above

     ```
     $ git clone https://github.com/marskar/snakemake.git
     ```
     OR manually download
  2. Navigate to the directory containing the environment file

     ```
     $ cd snakemake
     ```
  3. Create the environment

     ```
     $ conda env create -f environment.yml
     $ conda activate snakemake_class
     ```
  4. Open the jupyter notebook

     ```
     $ jupyter notebook
     ```
     - The notebook is at snakemake/Genomics_practice/Pipelines_for_genomics.ipynb



Helpful references

- Offical snakemake tutorial: https://snakemake.readthedocs.io/en/stable/tutorial/tutorial.html
- Beginner's tutorial: https://github.com/leipzig/SandwichesWithSnakemake
- UC Davis tutorial with binder: https://github.com/ctb/2019-snakemake-ucdavis/blob/master/tutorial.md
- Snakemake manuscript: https://academic.oup.com/bioinformatics/article/28/19/2520/290322
- Me: bari dot ballew at nih dot gov