Setup

```
bw$ sinteractive -c12 --mem=24g --gres=lscratch:20
...
node$ module load singularity snakemake hisat
node$ cd /data/$USER
node$ git clone https://github.com/NIH-HPC/snakemake-class.git
node$ cd snakemake-class
node$ ./setup.sh
...
+-------------------------------------------------+
|                                                 |
|  Class materials have been set up successfully  |
|                                                 |
+-------------------------------------------------+
```

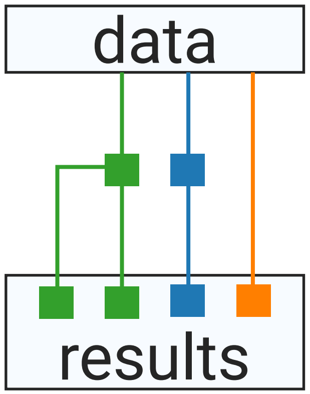
Building a reproducible workflow with Snakemake and Singularity

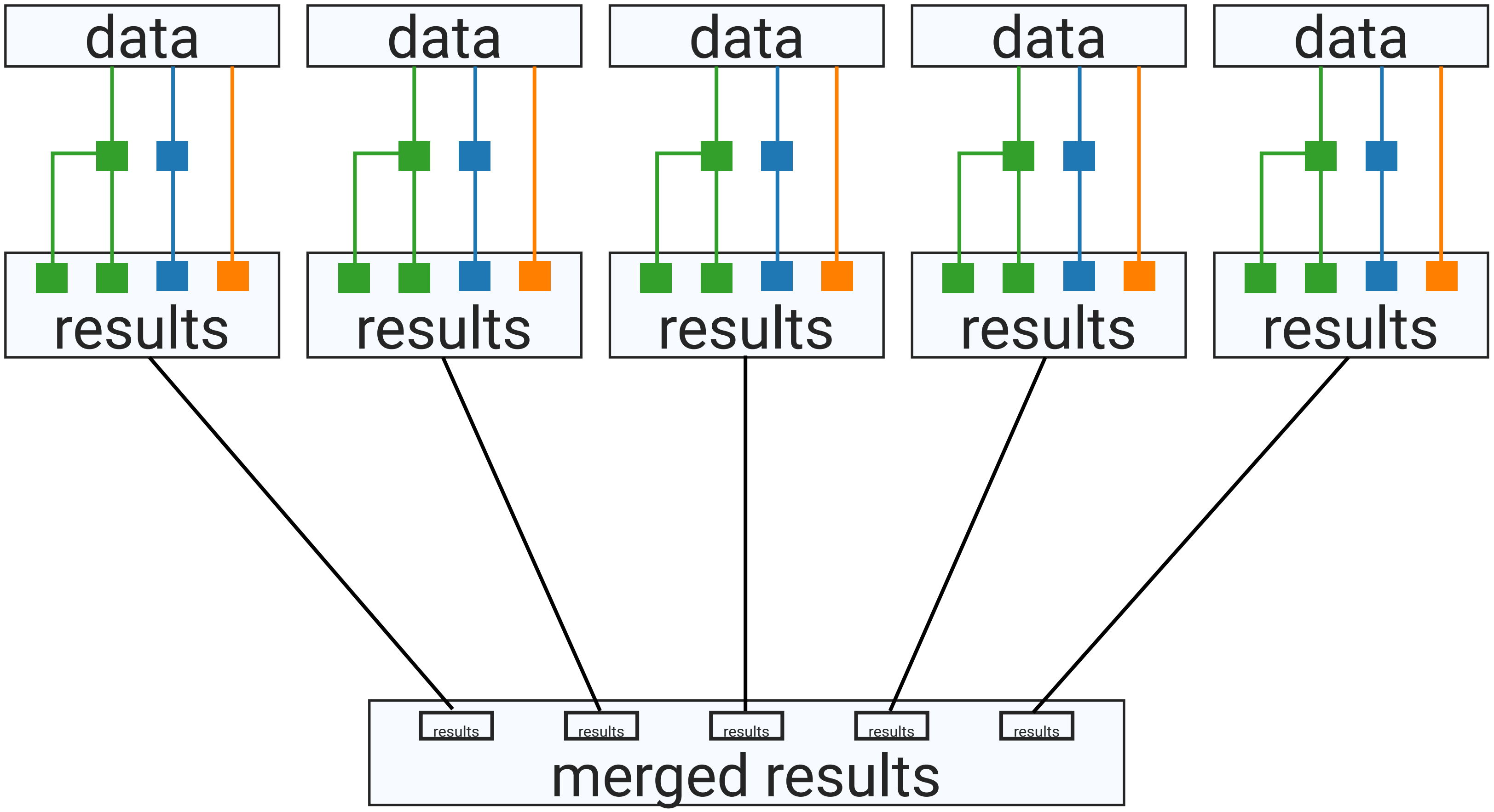
Wolfgang Resch - HPC @ NIH - staff@hpc.nih.gov

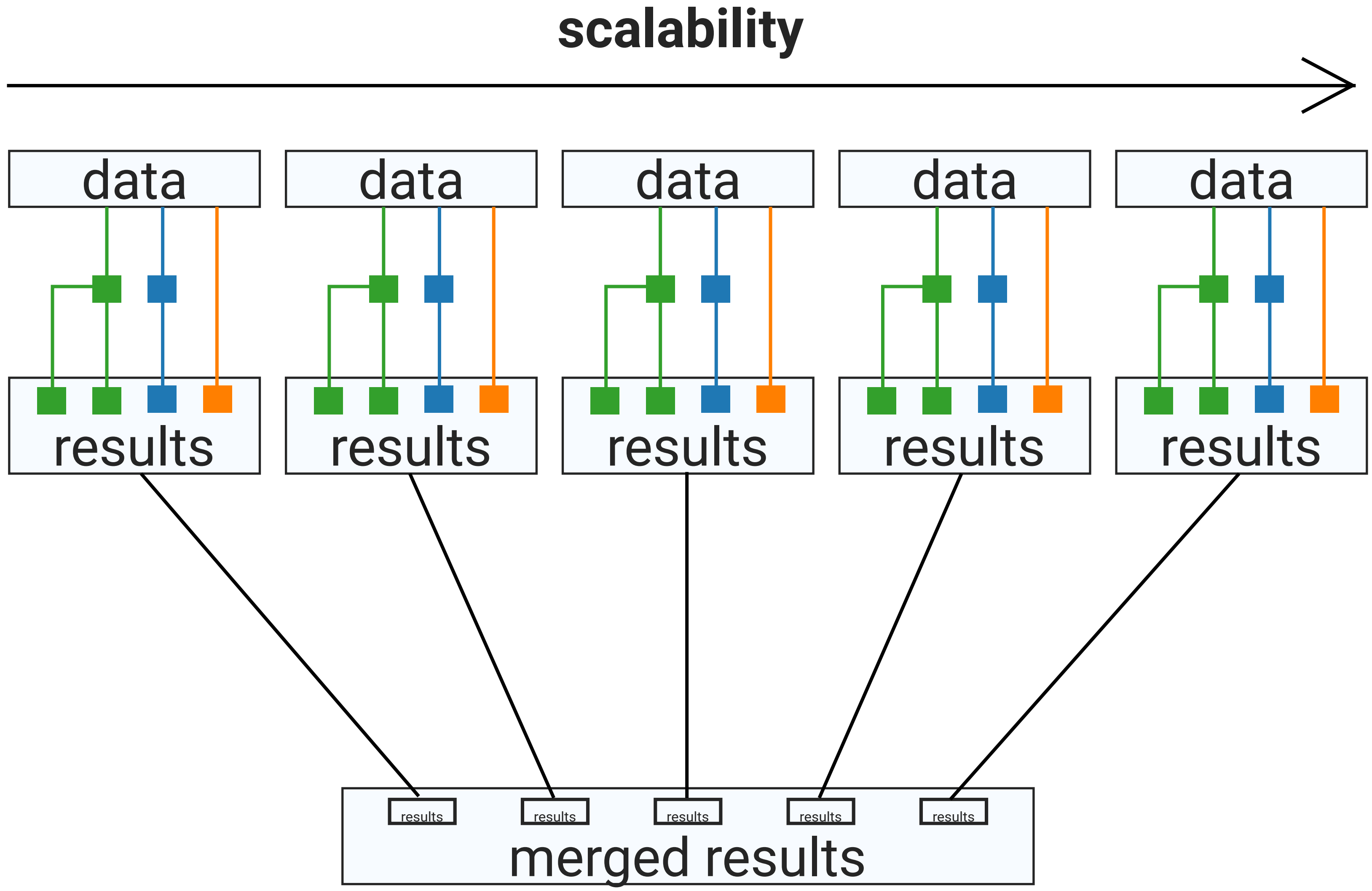
Slides adapted from

Johannes Koester

http://slides.com/johanneskoester/snakemake-tutorial-2016#/

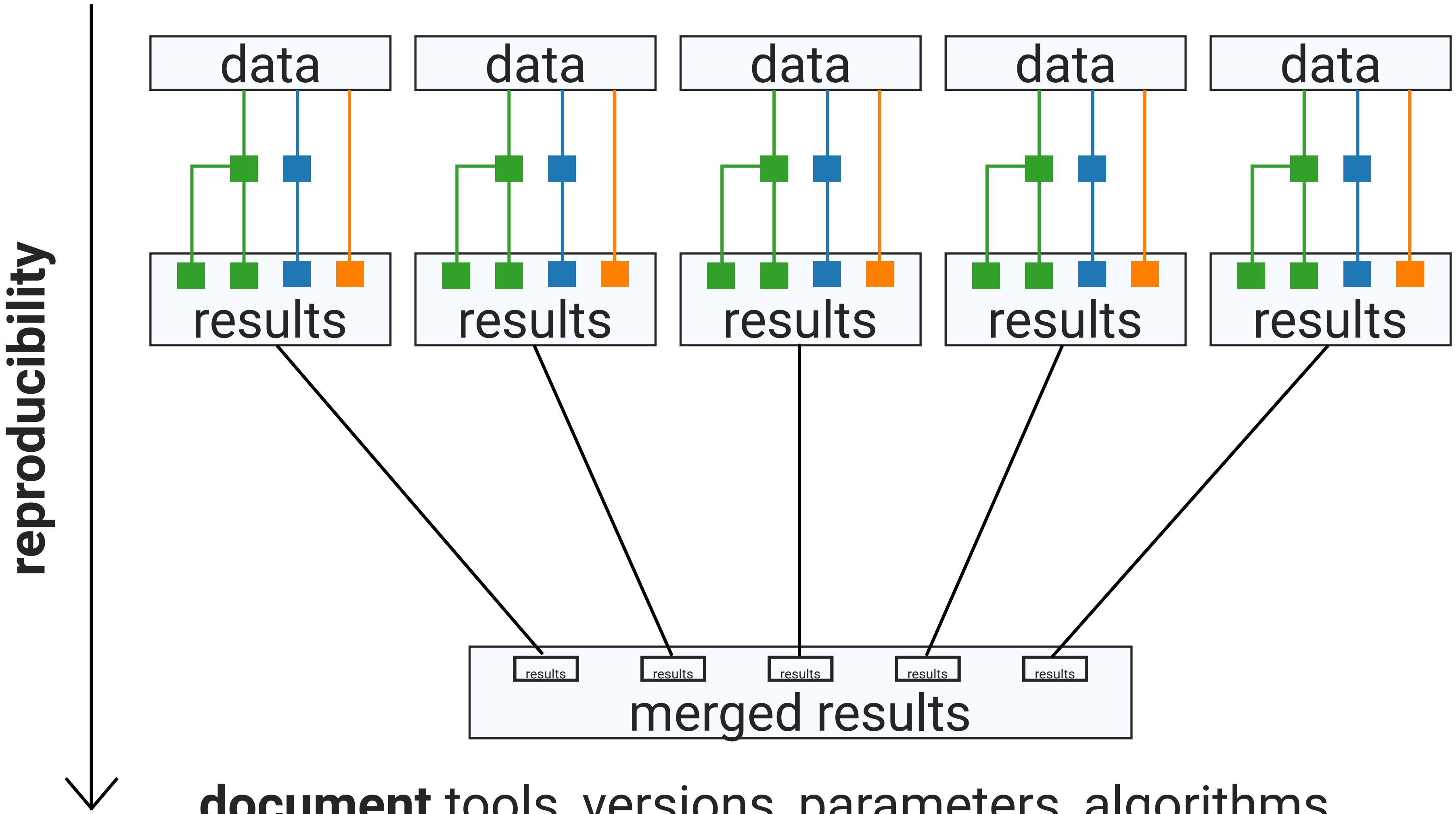

data
results


data
data
data
data
data
results
results
results
results
results
results
results
results
results
results
merged results



automatically execute steps in **parallel**

minimize redundant computation when adding/changing data, or resuming interrupted workflows



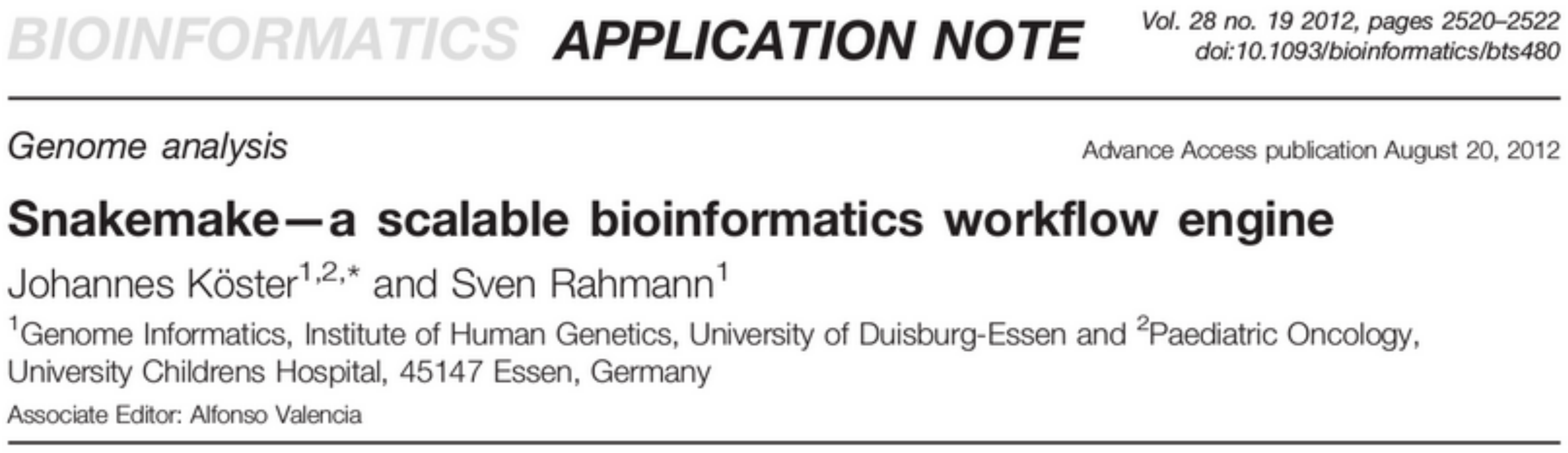
document tools, versions, parameters, algorithms
execute automatically

There Are Many Workflow Tools

make, ninja, scons, waf, ruffus, jug, Rake, bpipe, BigDataScript, toil, nextflow, paver, bcbio-nextgen, snakemake, *wdl, cwl*, Galaxy, KNIME,Taverna, Partek flow, DNAnexus, SevenBridges, Basespace

https://github.com/pditommaso/awesome-pipeline

Snakemake



BIOINFORMATICS APPLICATION NOTE

Vol. 28 no. 19 2012, pages 2520–2522
doi:10.1093/bioinformatics/bts480

Genome analysis

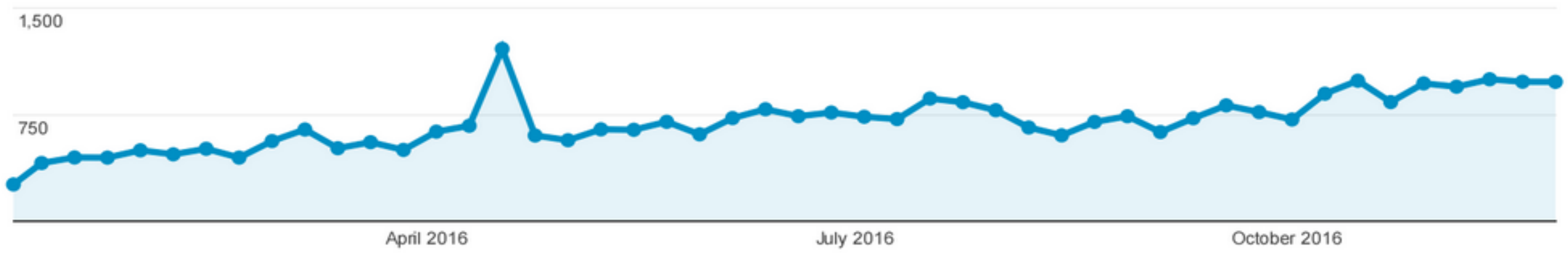
Advance Access publication August 20, 2012

Snakemake—a scalable bioinformatics workflow engine

Johannes Köster[1,2,*] and Sven Rahmann[1]

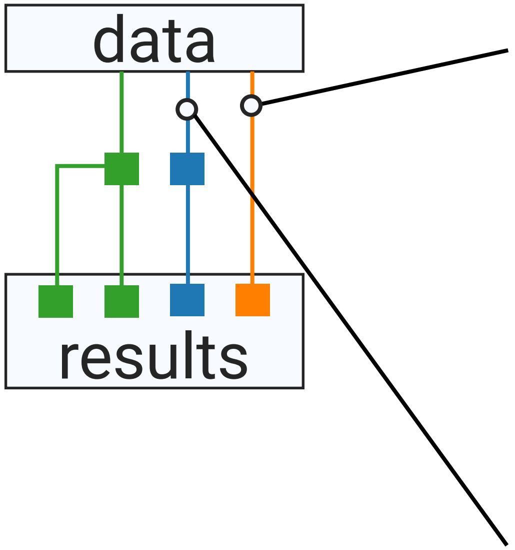
[1]Genome Informatics, Institute of Human Genetics, University of Duisburg-Essen and [2]Paediatric Oncology, University Childrens Hospital, 45147 Essen, Germany

Associate Editor: Alfonso Valencia



biowulf users: ~100

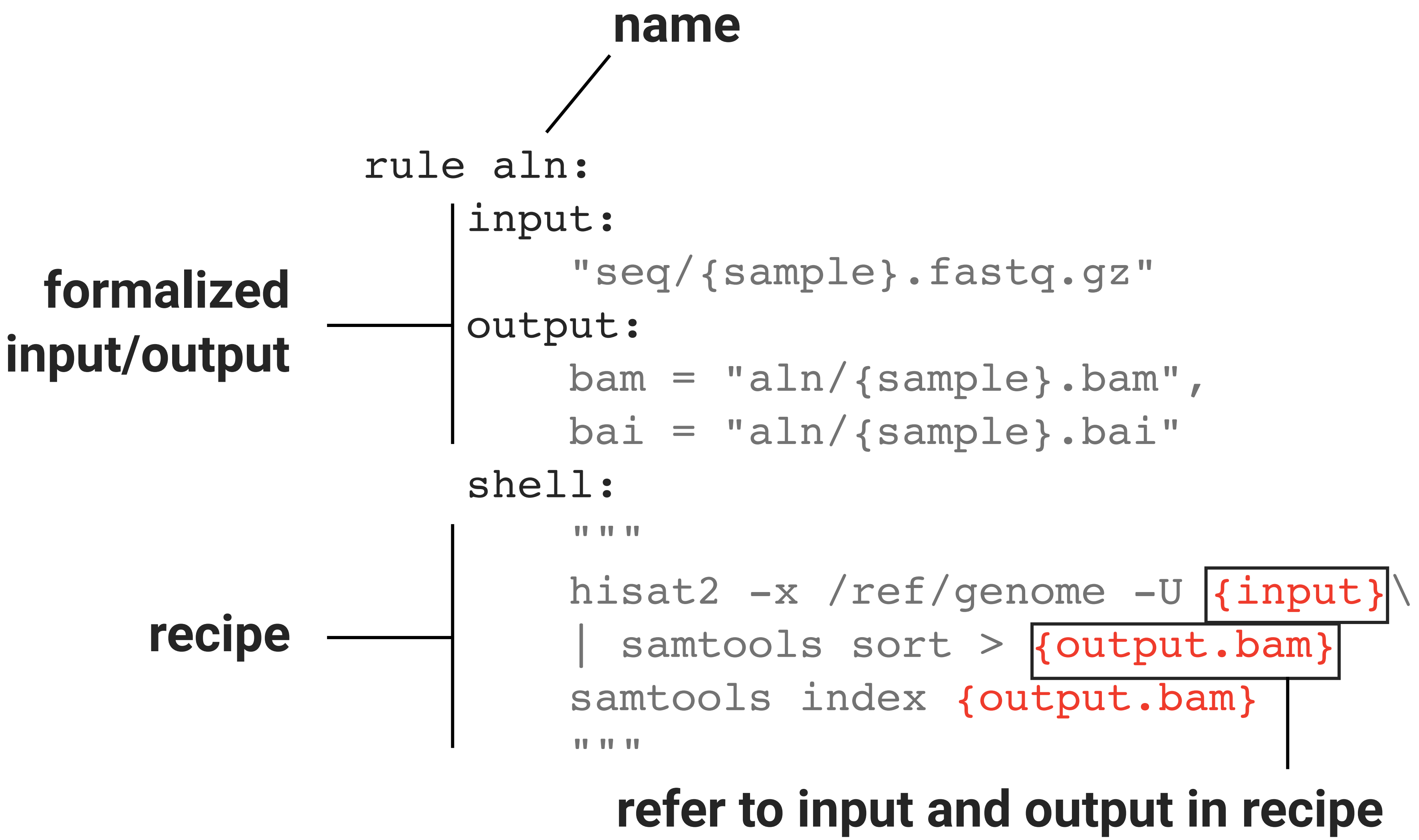
Rules



```
rule qc:
    input:
        "seq/{sample}.fastq.gz"
    output:
        "qc/{sample}.qc"
    script:
        "scripts/myscript.py"

rule aln:
    input:
        "seq/{sample}.fastq.gz"
    output:
        bam = "aln/{sample}.bam",
        bai = "aln/{sample}.bai"
    shell:
        """
        hisat2 -x /ref/genome -U {input}\
        | samtools sort > {output.bam}
        samtools index {output.bam}
        """
```

Rules



```
rule aln:
    input:
        "seq/{sample}.fastq.gz"
    output:
        bam = "aln/{sample}.bam",
        bai = "aln/{sample}.bai"
    shell:
        """
        hisat2 -x /ref/genome -U {input}\
        | samtools sort > {output.bam}
        samtools index {output.bam}
        """
```

Rules

wildcards generalize rules

```
rule aln:
    input:
        "seq/{sample}.fastq.gz"
    output:
        bam = "aln/{sample}.bam",
        bai = "aln/{sample}.bai"
    shell:
        """
        hisat2 -x /ref/genome -U {input}\
        | samtools sort > {output.bam}
        samtools index {output.bam}
        """
```

Rules

```
rule aln:
    input:
        "seq/{sample}.fastq.gz"
    output:
        bam = "aln/{sample}.bam",
        bai = "aln/{sample}.bai"
    shell:
        """
        hisat2 -x /ref/genome -U {input}\
        | samtools sort > {output.bam}
        samtools index {output.bam}
        """
```

input and output can be **single files** or **lists**

referred to **by name**

Rules

input and output can be **single files** or **lists**

```
rule aln:
    input:
        "seq/{sample}.fastq.gz"
    output:
        "aln/{sample}.bam",
        "aln/{sample}.bai"
    shell:
        """
        hisat2 -x /ref/genome -U {input}\
        | samtools sort > {output[0]}
        samtools index {output[0]}
        """
```

referred to **by index**

Rules

reproducible environment

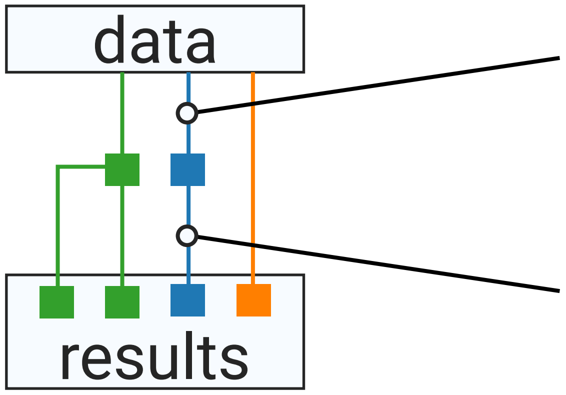
```
rule aln:
    input:
        "seq/{sample}.fastq.gz"
    output:
        "aln/{sample}.bam",
        "aln/{sample}.bai"
    conda:
        "envs/aln.yml"
    shell:
        """
        hisat2 -x /ref/genome -U {input}\
        | samtools sort > {output[0]}
        samtools index {output[0]}
        """
```

Rules

```
rule aln:
    input:
        "seq/{sample}.fastq.gz"
    output:
        "aln/{sample}.bam",
        "aln/{sample}.bai"
    singularity:
        "shub://NIH-HPC/snakemake-class"
    shell:
        """
        hisat2 -x /ref/genome -U {input}\
        | samtools sort > {output[0]}
        samtools index {output[0]}
        """
```

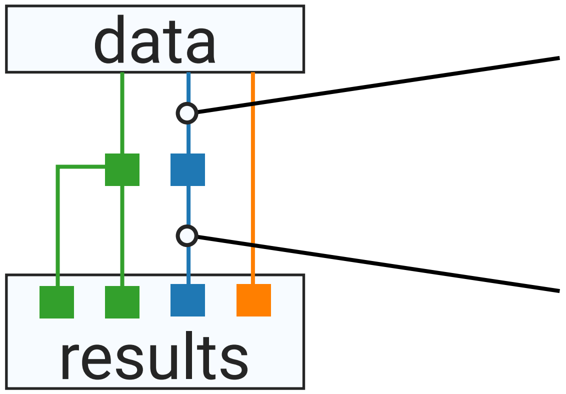
reproducible environment

Dependencies are implicit and 'backwards'



```
rule a:
    input: "start/{sample}.txt"
    output: "mid/{sample}.txt"
    shell: "sort {input} > {output}"
rule b:
    input: "mid/{sample}.txt"
    output: "final/{sample}.summary"
    shell: "uniq -c {input} > {output}"
```

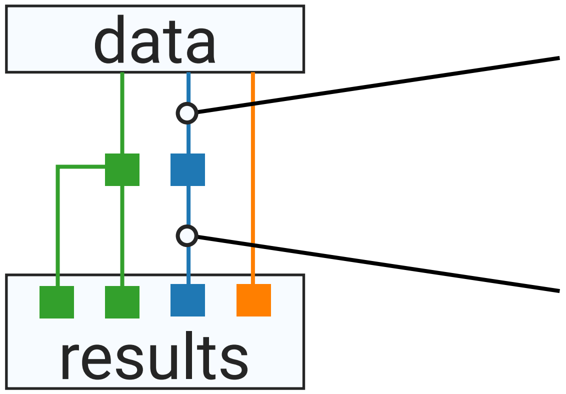
Dependencies are implicit and 'backwards'



```
rule a:
    input: "start/{sample}.txt"
    output: "mid/{sample}.txt"
    shell: "sort {input} > {output}"
rule b:
    input: "mid/{sample}.txt"
    output: "final/{sample}.summary"
    shell: "uniq -c {input} > {output}"
```

```
$ snakemake final/ABC.summary
```

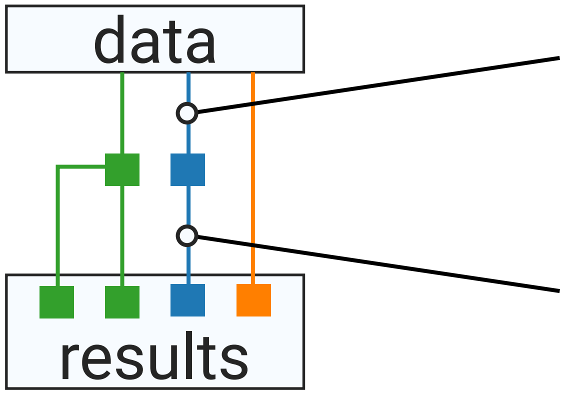
Dependencies are implicit and 'backwards'



```
rule a:
    input: "start/{sample}.txt"
    output: "mid/{sample}.txt"
    shell: "sort {input} > {output}"
rule b:
    input: "mid/{sample}.txt"
    output: "final/{sample}.summary"
    shell: "uniq -c {input} > {output}"
```

```
$ snakemake final/ABC.summary
```

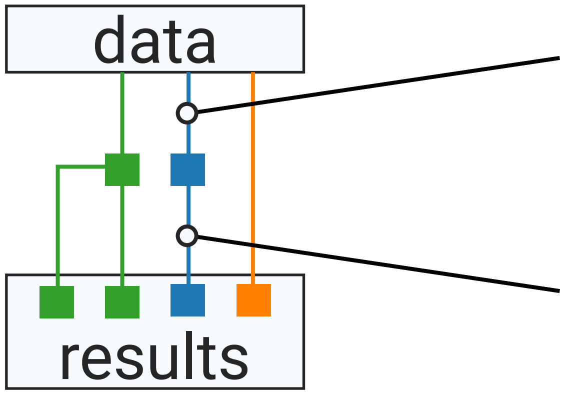
Dependencies are implicit and 'backwards'



```
rule a:
    input: "start/{sample}.txt"
    output: "mid/{sample}.txt"
    shell: "sort {input} > {output}"
rule b:
    input: "mid/{sample}.txt"
    output: "final/ ABC .summary"
    shell: "uniq -c {input} > {output}"
```

```
$ snakemake final/ABC.summary
```

Dependencies are implicit and 'backwards'

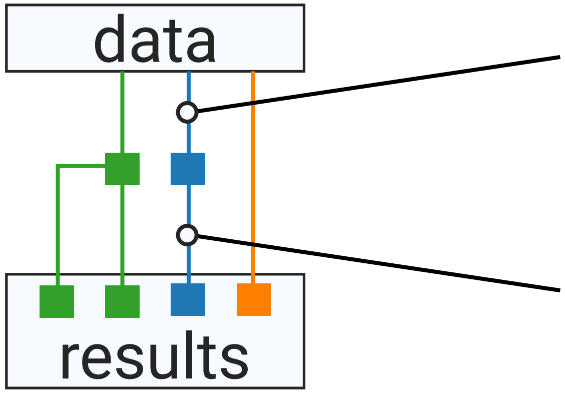


```
rule a:
    input: "start/{sample}.txt"
    output: "mid/{sample}.txt"
    shell: "sort {input} > {output}"

rule b:
    input: "mid/ ABC .txt"
    output: "final/ ABC .summary"
    shell: "uniq -c {input} > {output}"
```

```
$ snakemake final/ABC.summary
```

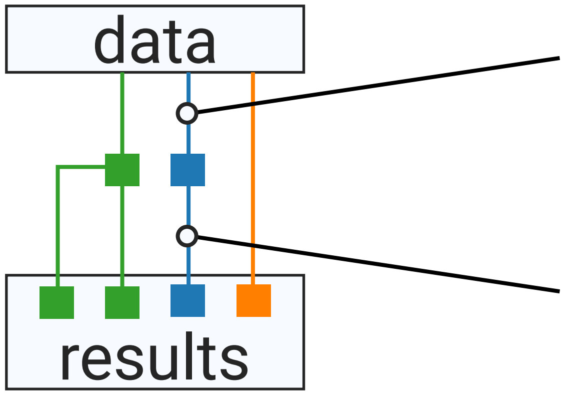
Dependencies are implicit and 'backwards'



```
rule a:
    input: "start/{sample}.txt"
    output: "mid/{sample}.txt"
    shell: "sort {input} > {output}"
rule b:
    input: "mid/ABC.txt"
    output: "final/ABC.summary"
    shell: "uniq -c {input} > {output}"
```

```
$ snakemake final/ABC.summary
```

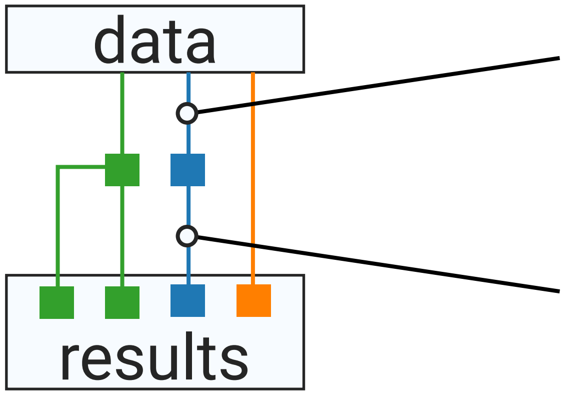
Dependencies are implicit and 'backwards'



```
rule a:
    input: "start/{sample}.txt"
    output: "mid/ABC.txt"
    shell: "sort {input} > {output}"
rule b:
    input: "mid/ABC.txt"
    output: "final/ABC.summary"
    shell: "uniq -c {input} > {output}"
```

```
$ snakemake final/ABC.summary
```

Dependencies are implicit and 'backwards'



```
rule a:
    input: "start/ABC.txt"
    output: "mid/ABC.txt"
    shell: "sort {input} > {output}"
rule b:
    input: "mid/ABC.txt"
    output: "final/ABC.summary"
    shell: "uniq -c {input} > {output}"
```

```
$ snakemake final/ABC.summary
```

http://slides.com/johanneskoester/snakemake-tutorial-2016#/

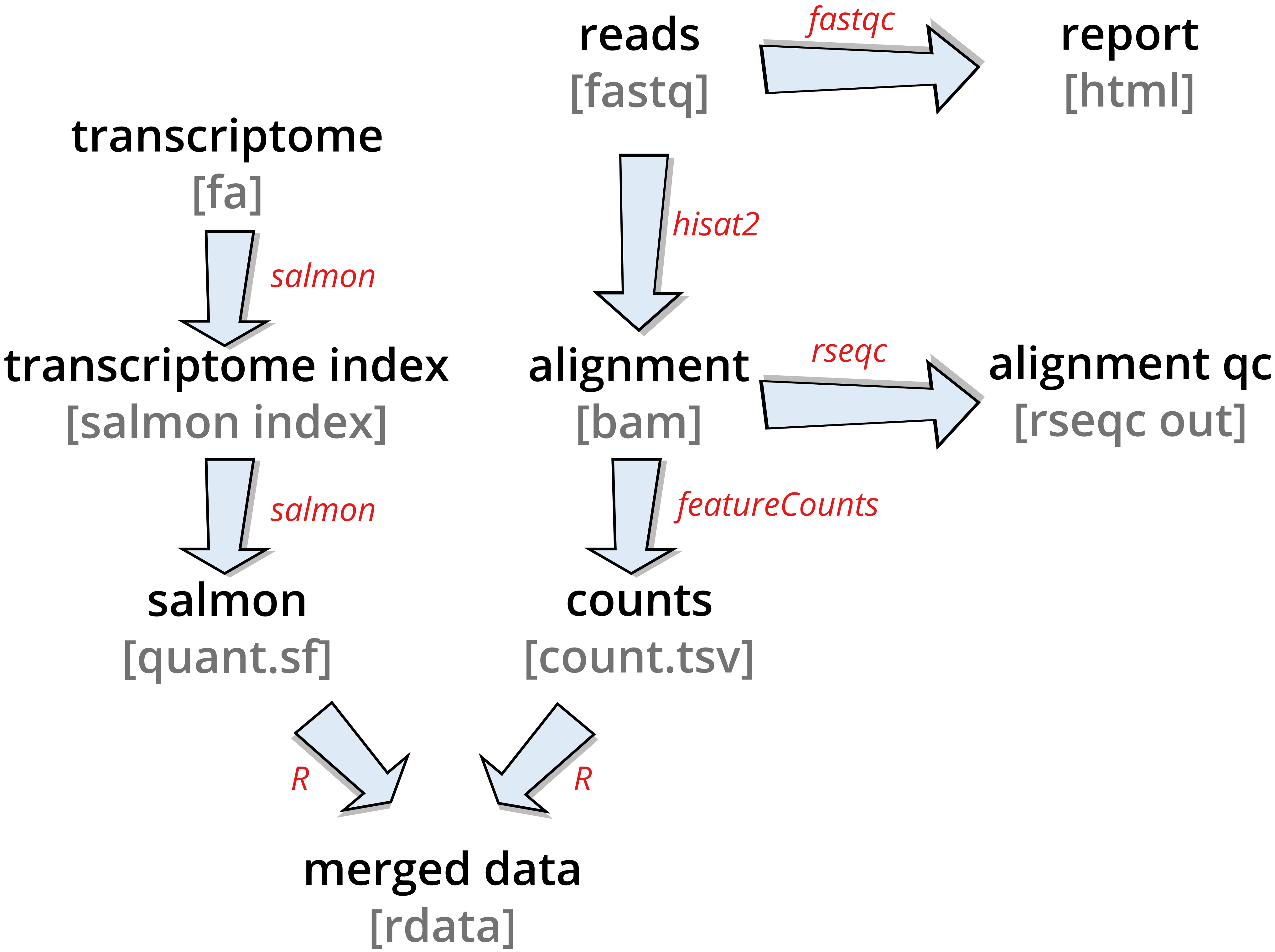
https://snakemake.readthedocs.io/en/stable/

https://bitbucket.org/snakemake/snakemake/overview

https://github.com/leipzig/SandwichesWithSnakemake

https://molb7621.github.io/workshop/Classes/snakemake-tutorial.html

http://blog.byronjsmith.com/snakemake-analysis.html


reads
[fastq]
fastqc
report
[html]
transcriptome
[fa]
hisat2
salmon
transcriptome index
[salmon index]
alignment
[bam]
rseqc
alignment qc
[rseqc out]
salmon
featureCounts
salmon
[quant.sf]
counts
[count.tsv]
R
R
merged data
[rdata]