# NIH Bring Your Own Bioinformatics (BYOB)

18 June 2020

# Bioinformatics Pipelines using JUDI: *Just Do It!*

Soumitra Pal – Research Fellow – Teresa M. Przytycka Lab


U.S. National Library of Medicine
National Center for Biotechnology Information

@soumitrakp

# Perhaps this is familiar to you …

# Perhaps this is familiar to you …

BI

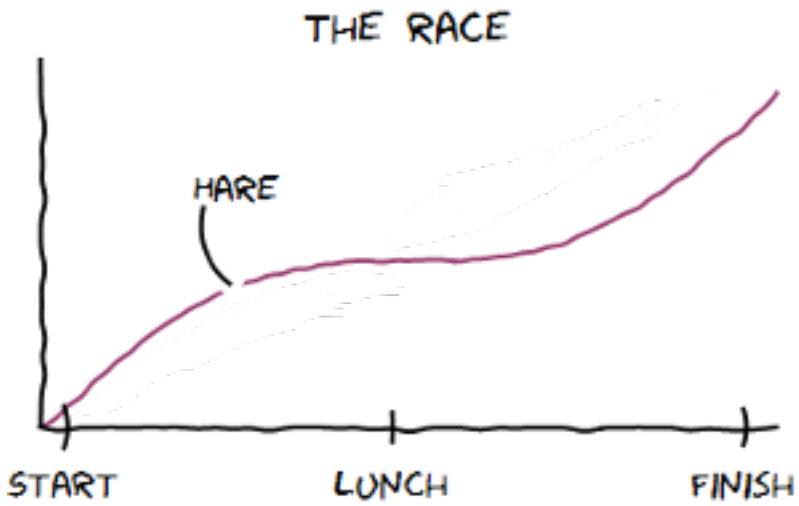
PI

# Perhaps this is familiar to you …

BI

PI

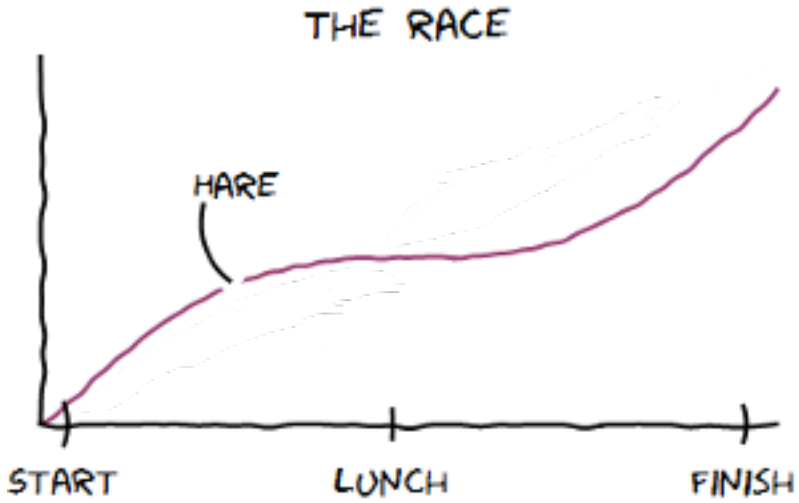
Here is the result



# Perhaps this is familiar to you …

BI

Here is the result




PI

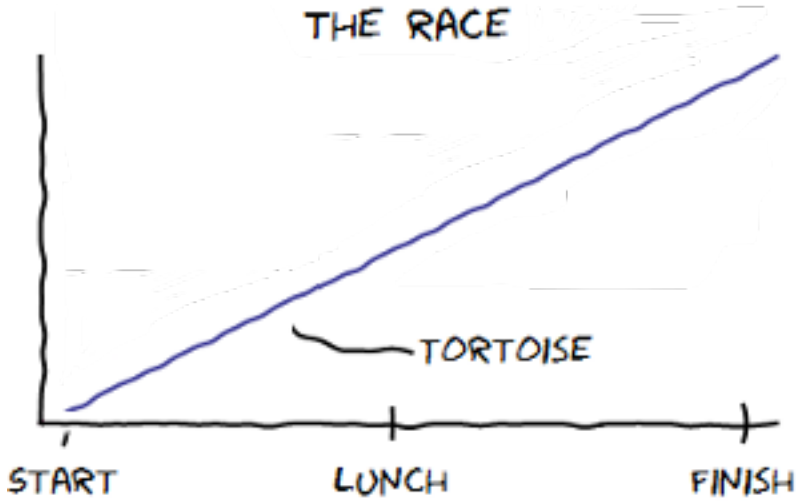
Looks good. Can you change racer (param X)?

# Perhaps this is familiar to you …

BI

PI

Here is the result




Looks good. Can you change racer (param X)?

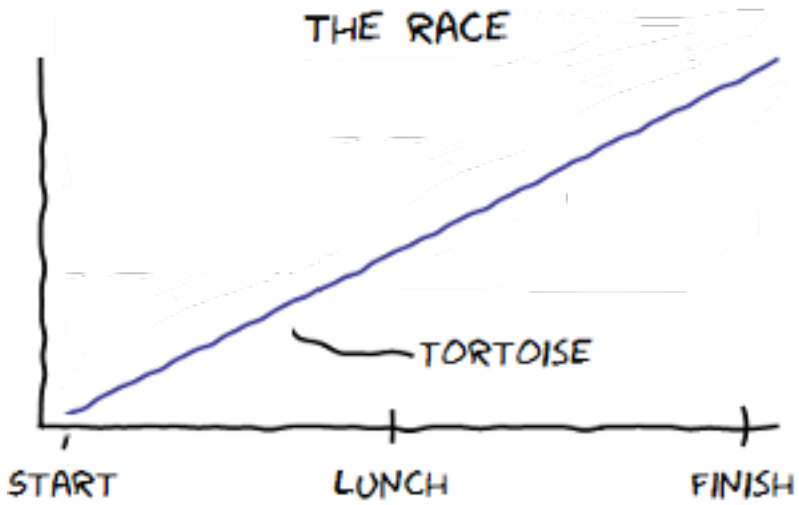
Ok, updated plot using a new value of X

# Perhaps this is familiar to you …

BI

Here is the result

Ok, updated plot using a new value of X




PI

Looks good. Can you change racer (param X)?

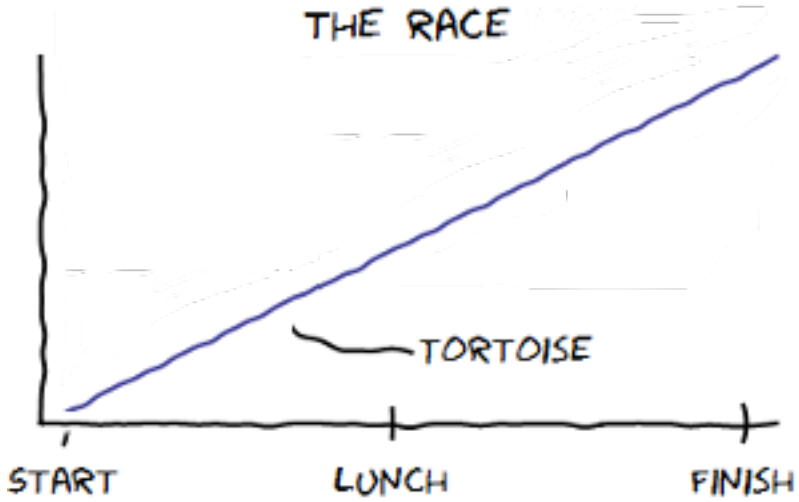
Better. Can u show plots for both values of X?

# Perhaps this is familiar to you …

BI

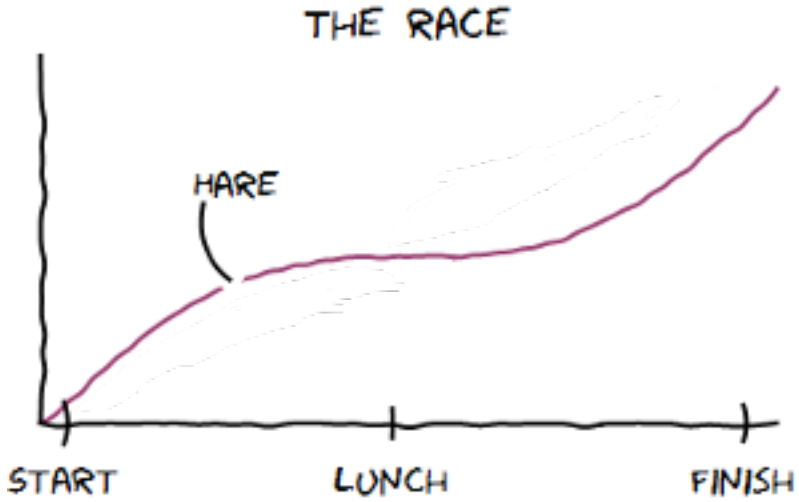
PI

Here is the result




Looks good. Can you change racer (param X)?

Ok, updated plot using a new value of X




Better. Can u show plots for both values of X?

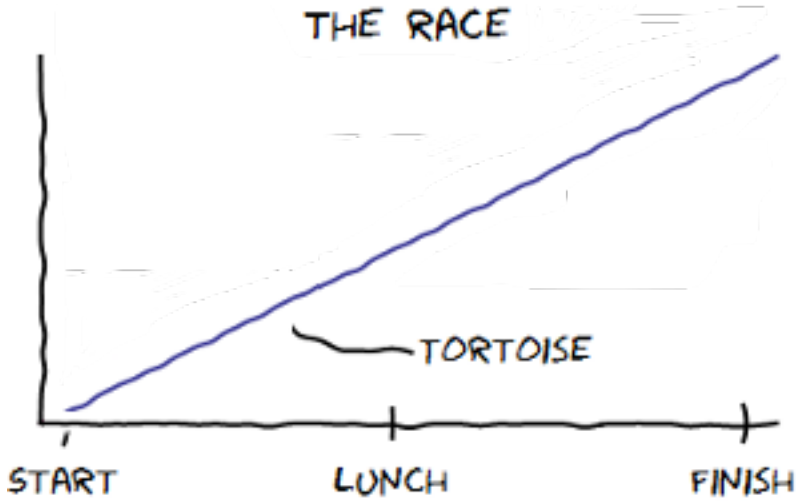
Ok, here are the two pdfs for the two plots

# Perhaps this is familiar to you …

BI

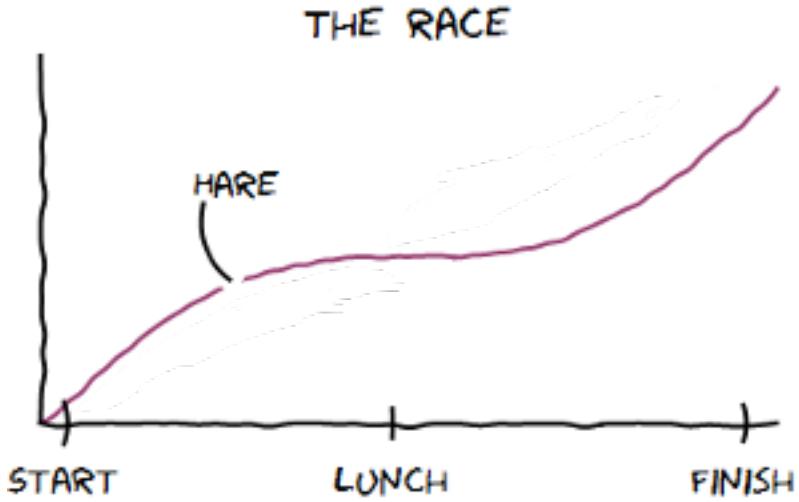
PI

Here is the result




Looks good. Can you change racer (param X)?

Ok, updated plot using a new value of X




Better. Can u show plots for both values of X?

Ok, here are the two pdfs for the two plots

Interesting. Can u show both curves in one plot?

# Perhaps this is familiar to you …

BI

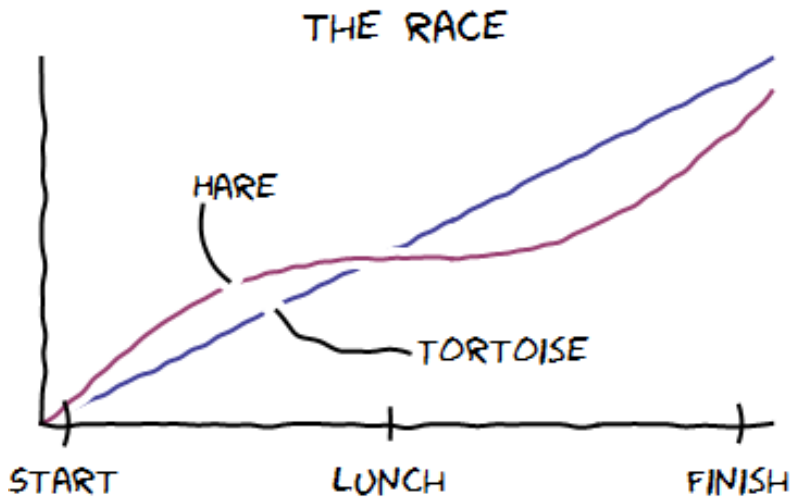
PI

Here is the result

Ok, updated plot using a new value of X

Ok, here are the two pdfs for the two plots

Ok, both in one plot




Looks good. Can you change racer (param X)?

Better. Can u show plots for both values of X?

Interesting. Can u show both curves in one plot?

# Perhaps this is familiar to you …

BI

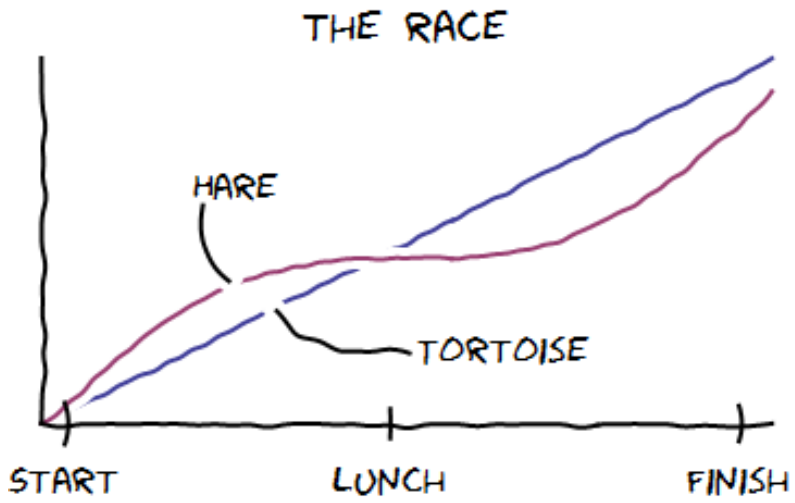
PI

Here is the result

Ok, updated plot using a new value of X

Ok, here are the two pdfs for the two plots

Ok, both in one plot




Looks good. Can you change racer (param X)?

Better. Can u show plots for both values of X?

Interesting. Can u show both curves in one plot?

And so on …

# This is a toy example, but imagine if …

| | Toy scenario | Real world example |
|---|---|---|

# This is a toy example, but imagine if …

| | Toy scenario | Real world example |
|---|---|---|
| Result | Plot | Could be any types of results:<br>- Table (DataFrame, Excel, …)<br>- Figure with multiple plots (pdf, png, …)<br>- Python,R,… objects to share with collaborators<br>- and so on |

# This is a toy example, but imagine if …

| | Toy scenario | Real world example |
|---|---|---|
| Result | Plot | Could be any types of results:<br>- Table (DataFrame, Excel, …)<br>- Figure with multiple plots (pdf, png, …)<br>- Python,R,… objects to share with collaborators<br>- and so on |
| Parameter | Racer | Could be any variability within your pipeline<br>- Sample type (tissue, sex, replicate, WT, …)<br>- Data analysis tool/algo (PCA,UMAP, …)<br>- Cutoffs (p-value, FDR, …)<br>- and so on |

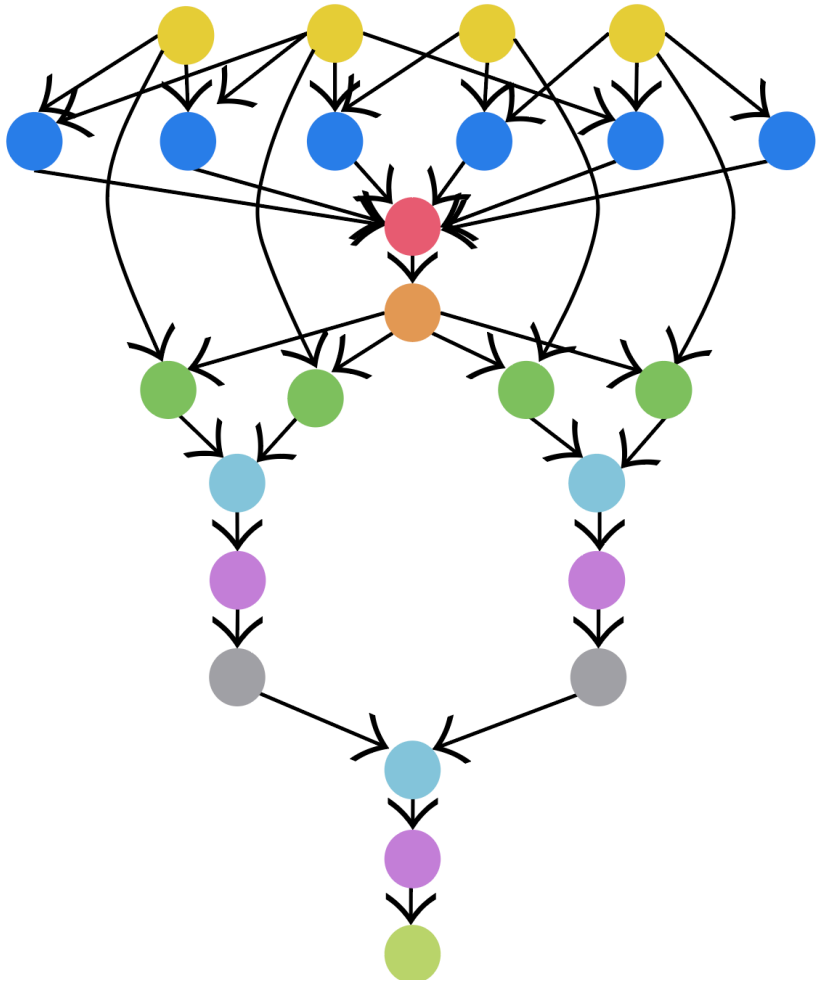
# This is a toy example, but imagine if …

| | Toy scenario | Real world example |
|---|---|---|
| Result | Plot | Could be any types of results:<br>- Table (DataFrame, Excel, …)<br>- Figure with multiple plots (pdf, png, …)<br>- Python,R,… objects to share with collaborators<br>- and so on |
| Parameter | Racer | Could be any variability within your pipeline<br>- Sample type (tissue, sex, replicate, WT, …)<br>- Data analysis tool/algo (PCA,UMAP, …)<br>- Cutoffs (p-value, FDR, …)<br>- and so on |
| Accumulation | 2-page PDF / 1-page combined | Could be any kind of aggregation<br>- Bigger DataFrame, Combined plot, …<br>- Merged file (Zip, Hd5, …)<br>- and so on |

# Moreover, imagine if …

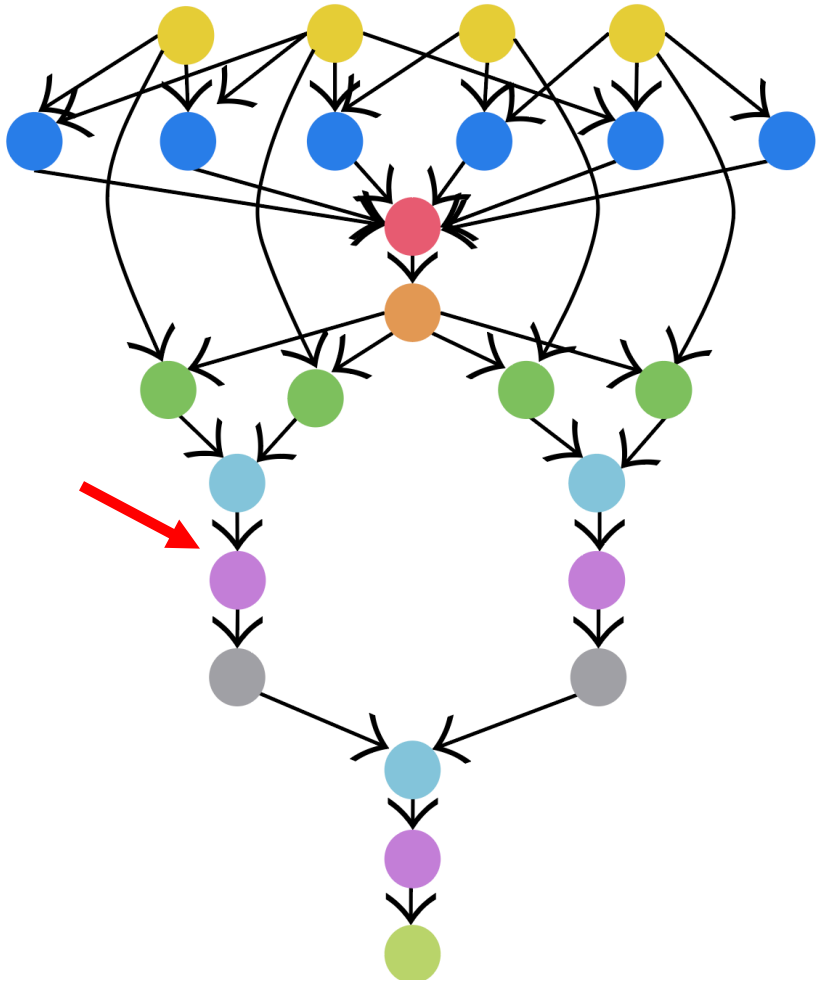
# Moreover, imagine if …

- Your pipeline is complex



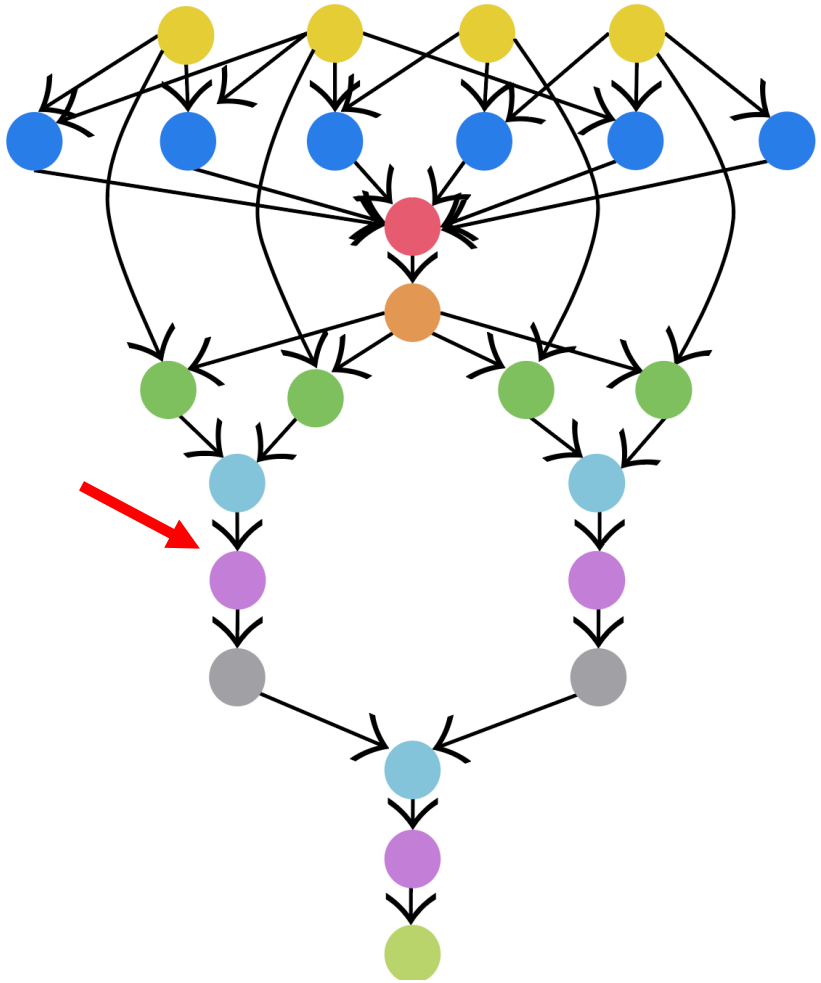
# Moreover, imagine if …

- Your pipeline is complex
- A parameter affects some stage deep within



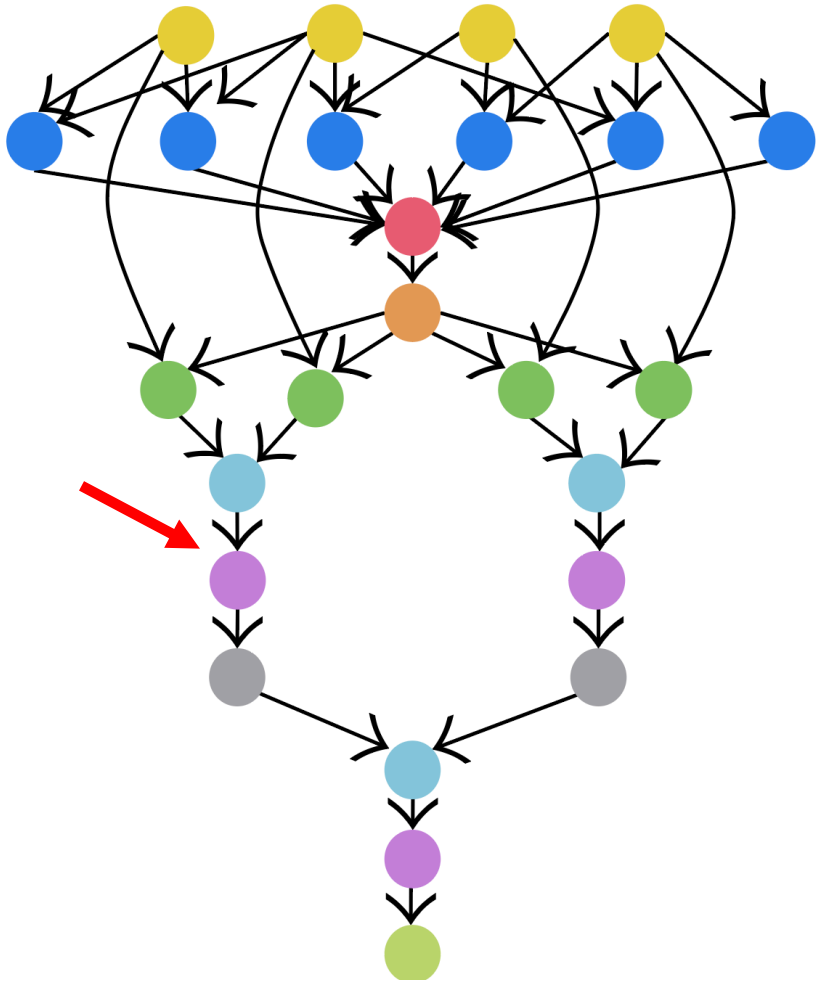
# Moreover, imagine if …

- Your pipeline is complex
- A parameter affects some stage deep within
- Unaffected stages need not re-execute



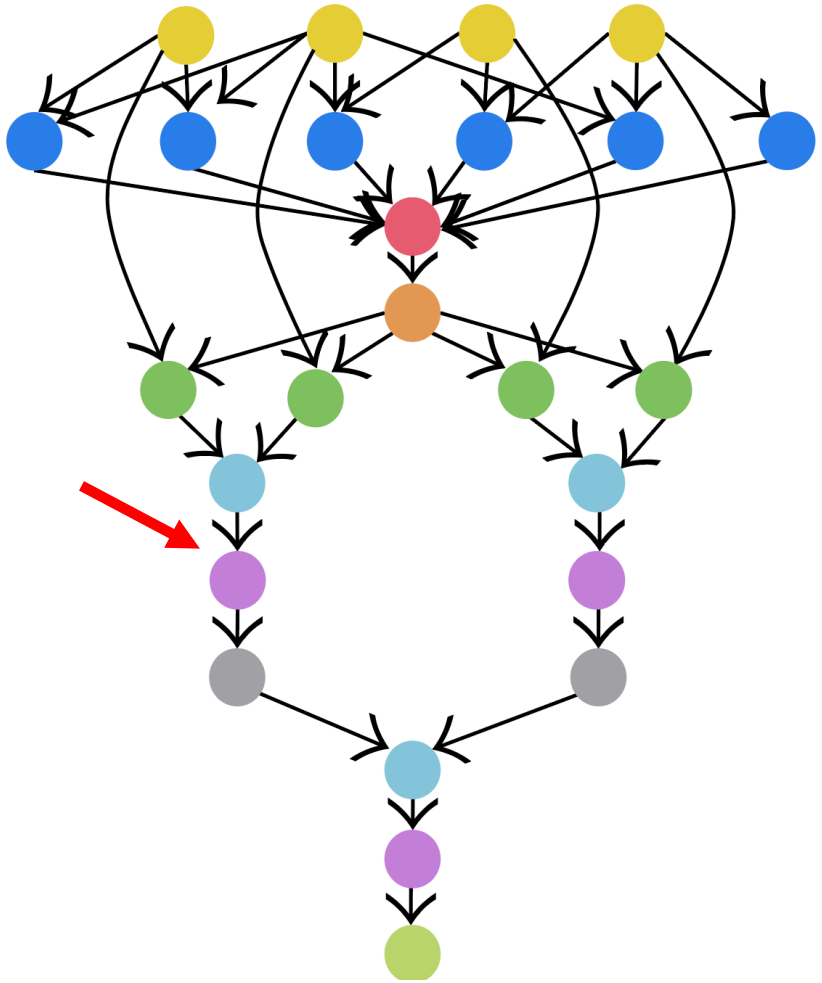
# Moreover, imagine if …

- Your pipeline is complex
- A parameter affects some stage deep within
- Unaffected stages need not re-execute
- Ensuring optimal execution under changes needs planning and effort



# Moreover, imagine if …

- Your pipeline is complex
- A parameter affects some stage deep within
- Unaffected stages need not re-execute
- Ensuring optimal execution under changes needs planning and effort
- Moreover, gathering of results under different parameter setting is also needed



# Need for a workflow manager

# Need for a workflow manager

Not only provide a way for optimal execution of tasks to produce output files from input files based on dependency but also:

# Need for a workflow manager

Not only provide a way for optimal execution of tasks to produce output files from input files based on dependency but also:

- Provide a way to capture variability under which workflow is executed

# Need for a workflow manager

Not only provide a way for optimal execution of tasks to produce output files from input files based on dependency but also:

- Provide a way to capture variability under which workflow is executed
- Optimal execution under different parameter settings

# Need for a workflow manager

Not only provide a way for optimal execution of tasks to produce output files from input files based on dependency but also:

- Provide a way to capture variability under which workflow is executed
- Optimal execution under different parameter settings
- Way to collate results under different parameter settings

# Existing workflow management systems (WMS)

# Existing workflow management systems (WMS)

- To our knowledge, no existing WMS supports all three requirements

# Existing workflow management systems (WMS)

- To our knowledge, no existing WMS supports all three requirements
- Looked three existing WMS in details

# Existing workflow management systems (WMS)

- To our knowledge, no existing WMS supports all three requirements
- Looked three existing WMS in details
- Snakemake [J Köster, S Rahmann, Bioinformatics, 2012]
  - Provides a way of parameter setting using wildcard but not rich enough

# Existing workflow management systems (WMS)

- To our knowledge, no existing WMS supports all three requirements
- Looked three existing WMS in details
- Snakemake [J Köster, S Rahmann, Bioinformatics, 2012]
  - Provides a way of parameter setting using wildcard but not rich enough
- Nextflow [Di Tommaso et al., Nat. Biotech. 2017]
  - Parameter settings require complex coding in a new language

# Existing workflow management systems (WMS)

- To our knowledge, no existing WMS supports all three requirements
- Looked three existing WMS in details
- Snakemake [J Köster, S Rahmann, Bioinformatics, 2012]
  - Provides a way of parameter setting using wildcard but not rich enough
- Nextflow [Di Tommaso et al., Nat. Biotech. 2017]
  - Parameter settings require complex coding in a new language
- DoIt [https://pydoit.org/]
  - Built on basic Python but doesn't support any special parameter setting

# Existing workflow management systems (WMS)

- To our knowledge, no existing WMS supports all three requirements
- Looked three existing WMS in details
- Snakemake [J Köster, S Rahmann, Bioinformatics, 2012]
  - Provides a way of parameter setting using wildcard but not rich enough
- Nextflow [Di Tommaso et al., Nat. Biotech. 2017]
  - Parameter settings require complex coding in a new language
- DoIt [https://pydoit.org/]
  - Built on basic Python but doesn't support any special parameter setting
- Decided to build a new system based on DoIt
  - DoIt has task:subtask structure could help different param settings
  - No new syntax than standard Python

# Existing workflow management systems (WMS)

- To our knowledge, no existing WMS supports all three requirements
- Looked three existing WMS in details
- Snakemake [J Köster, S Rahmann, Bioinformatics, 2012]
  - Provides a way of parameter setting using wildcard but not rich enough
- Nextflow [Di Tommaso et al., Nat. Biotech. 2017]
  - Parameter settings require complex coding in a new language
- DoIt [https://pydoit.org/]
  - Built on basic Python but doesn't support any special parameter setting
- Decided to build a new system based on DoIt
  - DoIt has task:subtask structure could help different param settings
  - No new syntax than standard Python

**JUDI = Just DoIt**

# Outline of the rest of the talk

# Outline of the rest of the talk

- Step-by-step introduction to key JUDI concepts using toy example

# Outline of the rest of the talk

- Step-by-step introduction to key JUDI concepts using toy example
- A more realistic example pipeline using JUDI concepts

# Outline of the rest of the talk

- Step-by-step introduction to key JUDI concepts using toy example
- A more realistic example pipeline using JUDI concepts
- Python code the build the pipeline using JUDI

# Outline of the rest of the talk

- Step-by-step introduction to key JUDI concepts using toy example
- A more realistic example pipeline using JUDI concepts
- Python code the build the pipeline using JUDI
- How to execute and get information about the pipeline

# Outline of the rest of the talk

- Step-by-step introduction to key JUDI concepts using toy example
- A more realistic example pipeline using JUDI concepts
- Python code the build the pipeline using JUDI
- How to execute and get information about the pipeline
- How to execute pipeline in multiprocessing env including Biowulf

# Outline of the rest of the talk

- Step-by-step introduction to key JUDI concepts using toy example
- A more realistic example pipeline using JUDI concepts
- Python code the build the pipeline using JUDI
- How to execute and get information about the pipeline
- How to execute pipeline in multiprocessing env including Biowulf
- Some comparison with Snakemake on the features

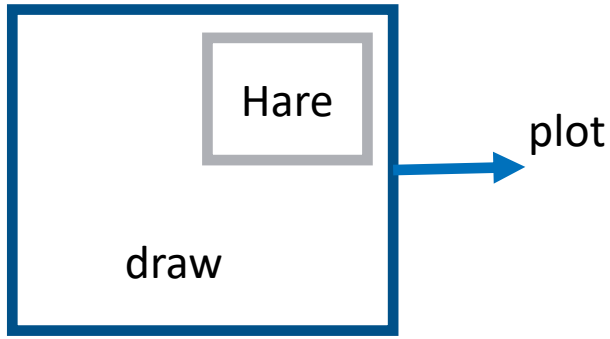
# Outline of the rest of the talk

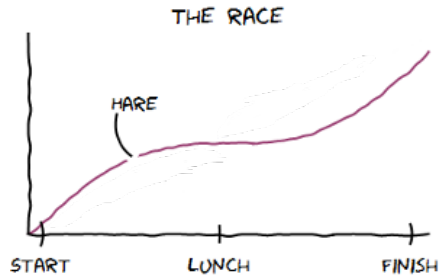
- Step-by-step introduction to key JUDI concepts using toy example
- A more realistic example pipeline using JUDI concepts
- Python code the build the pipeline using JUDI
- How to execute and get information about the pipeline
- How to execute pipeline in multiprocessing env including Biowulf
- Some comparison with Snakemake on the features
- Conclude with availability details

# Revisit toy pipeline: manual build single plot

# Revisit toy pipeline: manual build single plot

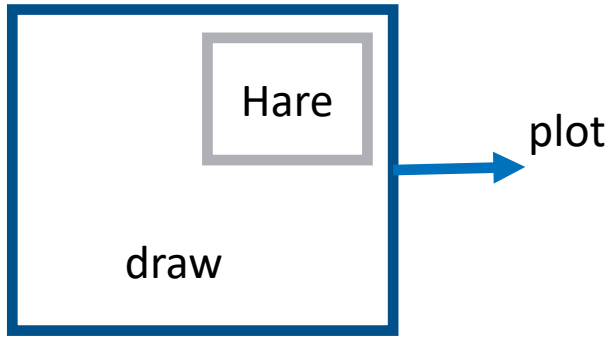
| time | dist |
|---|---|
| START | 0 |
| 1 | 0.25 |
| … | |
| LUNCH | 0.5 |
| … | |
| FINISH | 0.9 |

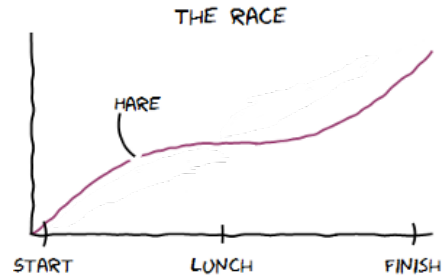


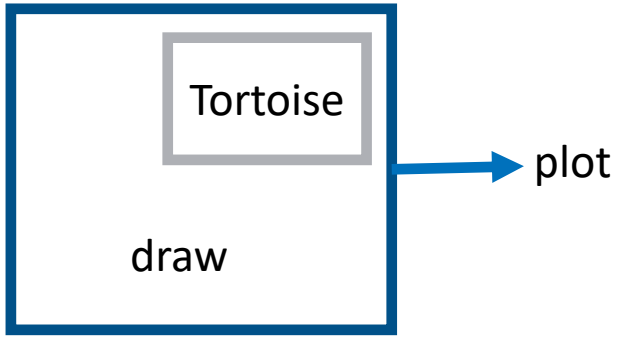
# Revisit toy pipeline: manual build single plot

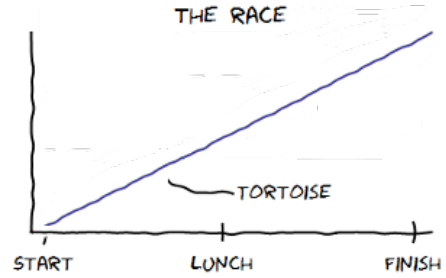
| time | dist |
|---|---|
| START | 0 |
| 1 | 0.25 |
| … | |
| LUNCH | 0.5 |
| … | |
| FINISH | 0.9 |



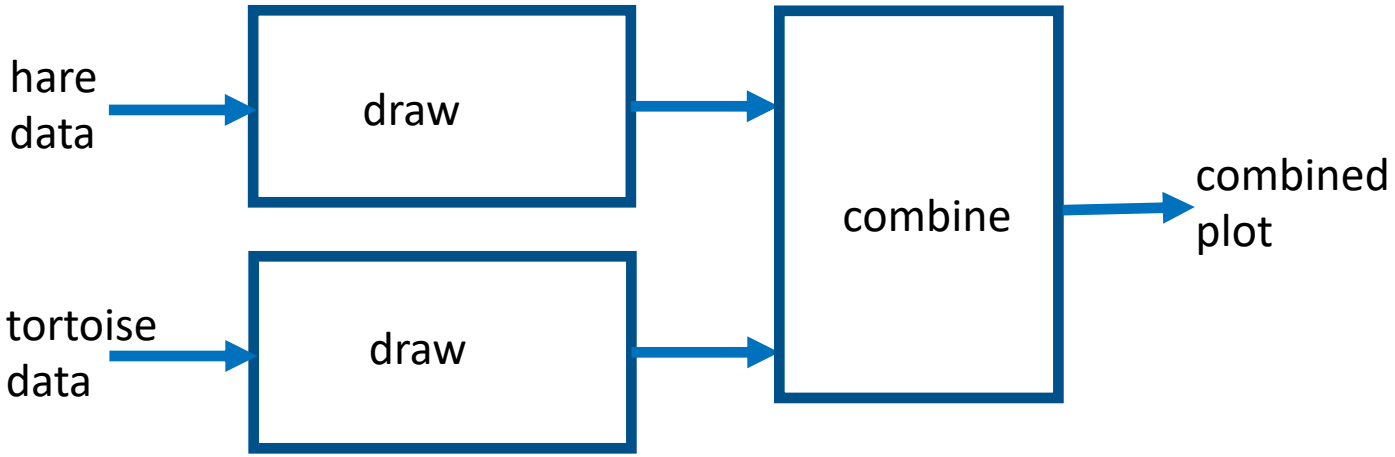

| time | dist |
|---|---|
| START | 0 |
| 1 | 0.1 |
| … | |
| LUNCH | 0.5 |
| … | |
| FINISH | 1 |



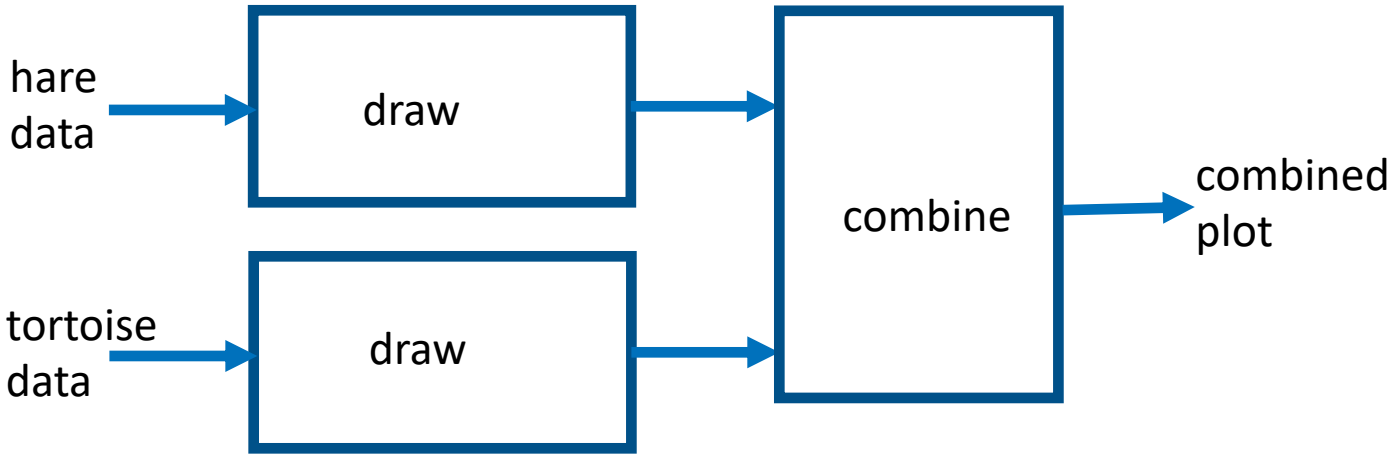



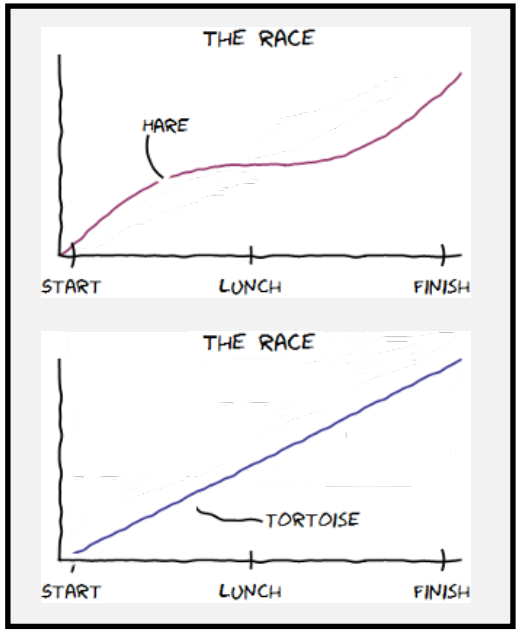
# Revisit toy pipeline: both plots by modularization

# Revisit toy pipeline: both plots by modularization

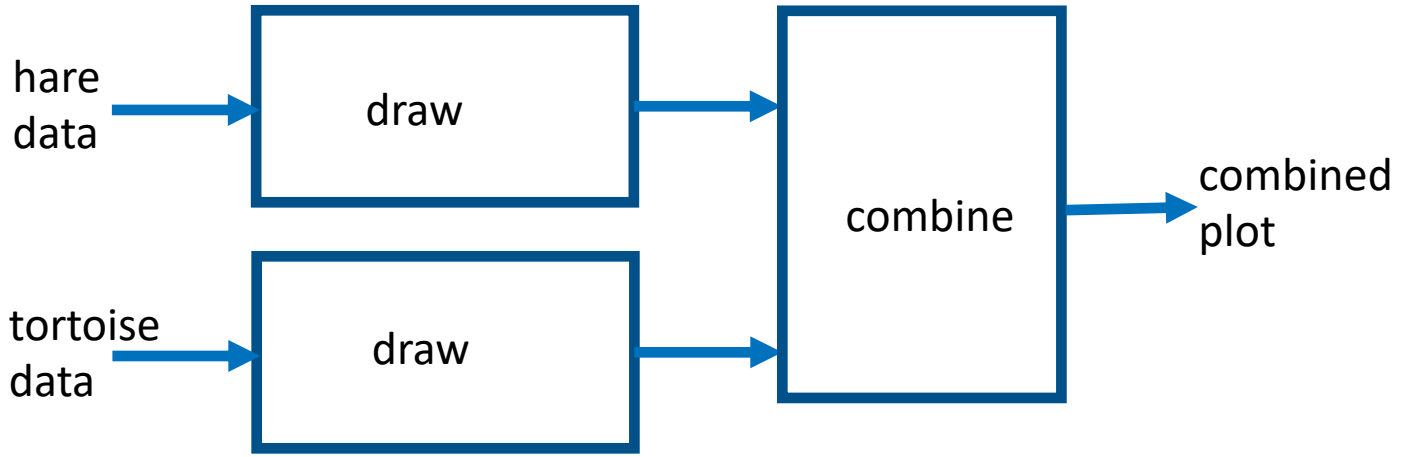


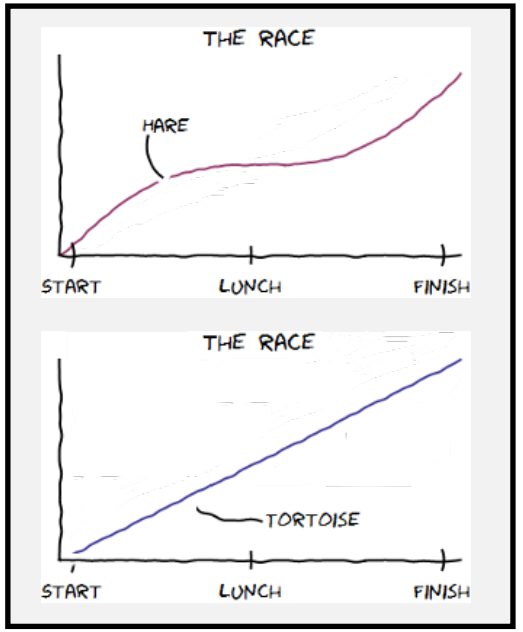
# Revisit toy pipeline: both plots by modularization

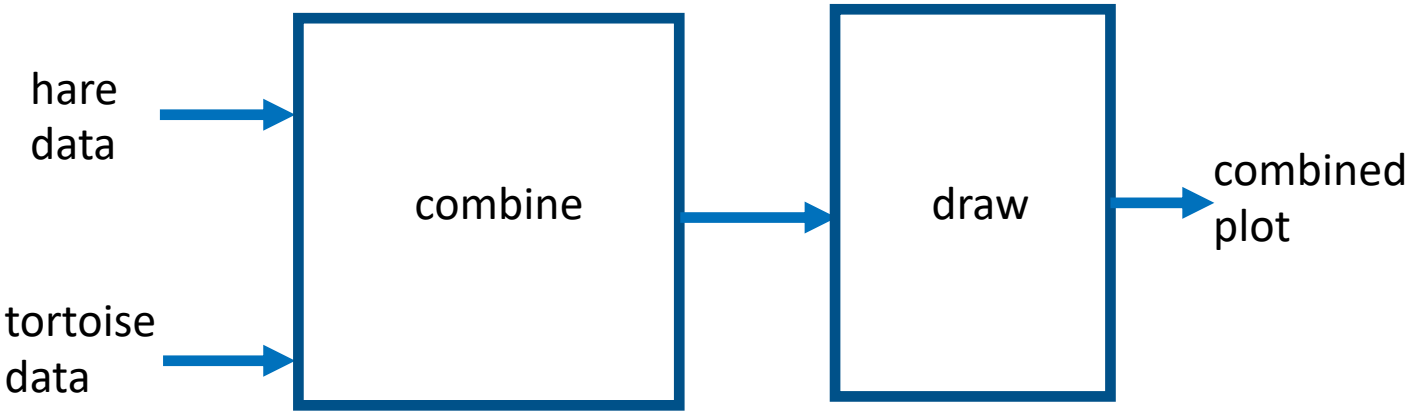


# Revisit toy pipeline: both plots by modularization

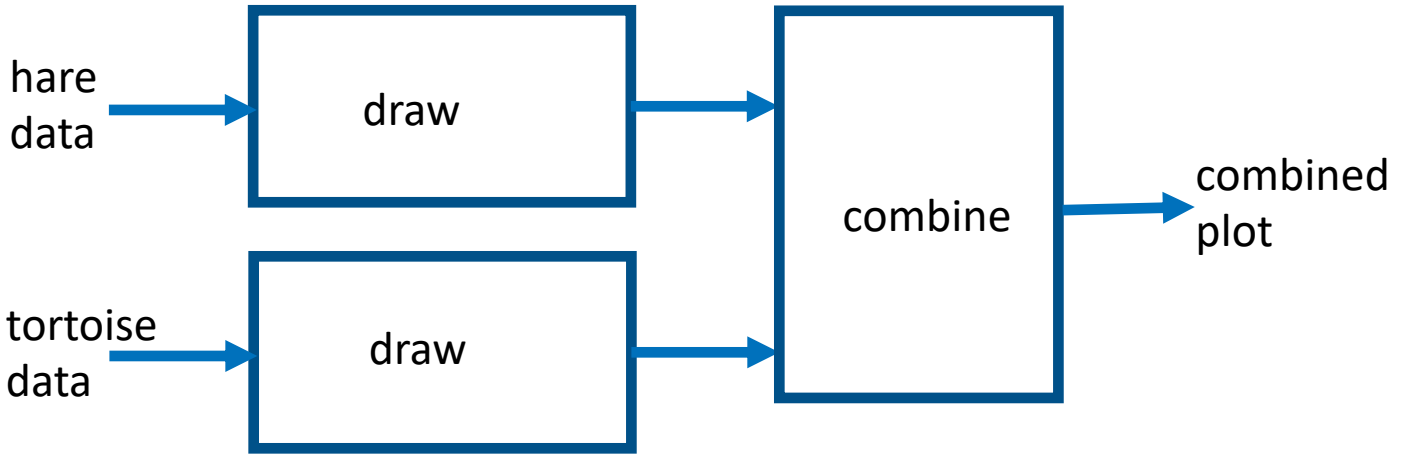


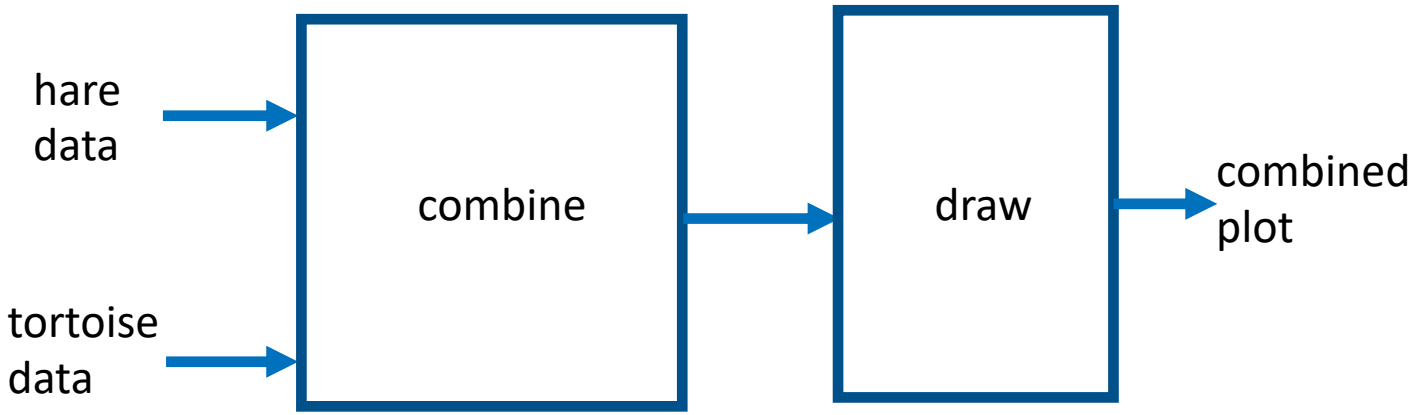



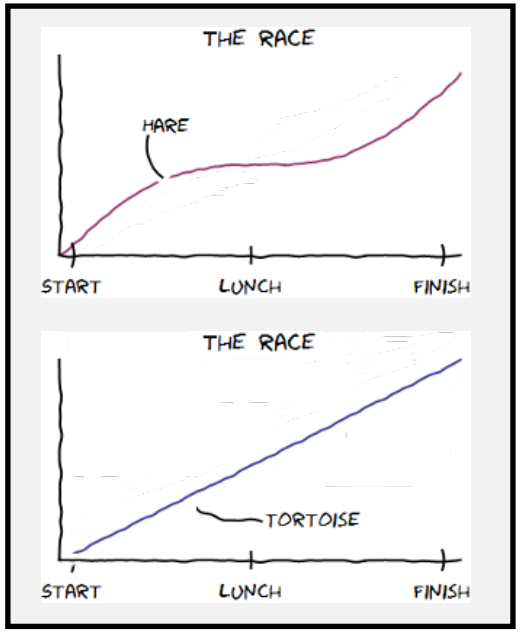



# Revisit toy pipeline: both plots by modularization

| racer | time | dist |
|---|---|---|
| hare | START | 0 |
| hare | 1 | 0.25 |
| | … | |
| hare | LUNCH | 0.5 |
| | … | |
| hare | FINISH | 0.9 |
| tort | START | 0 |
| tort | 1 | 0.1 |
| | … | |
| tort | LUNCH | 0.5 |
| | … | |
| tort | FINISH | 1 |

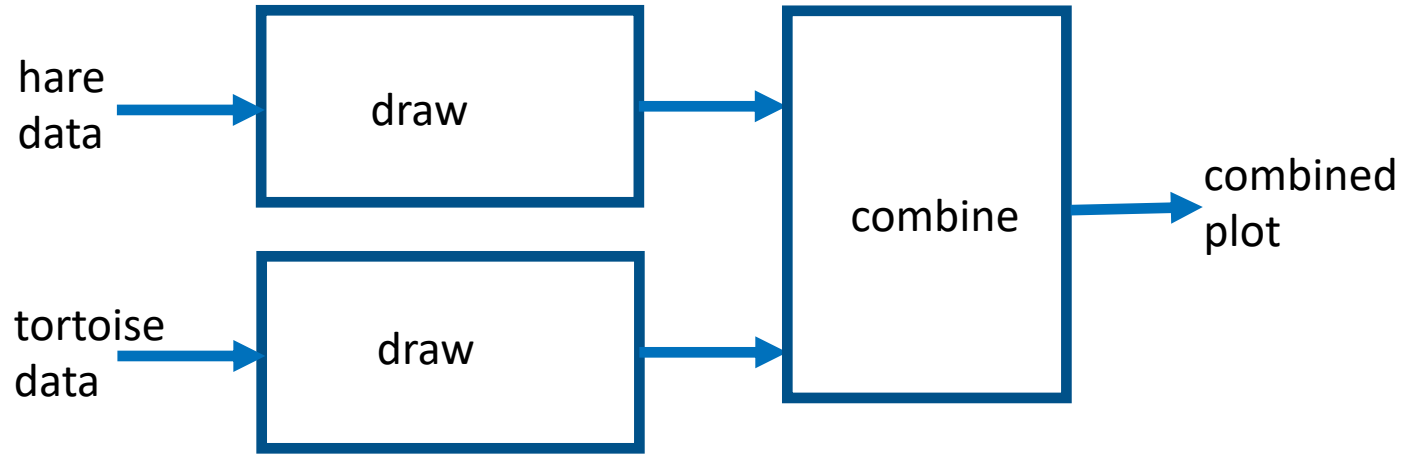


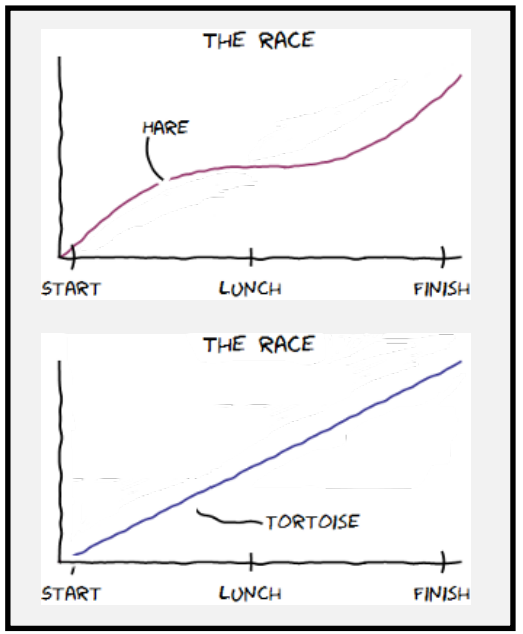



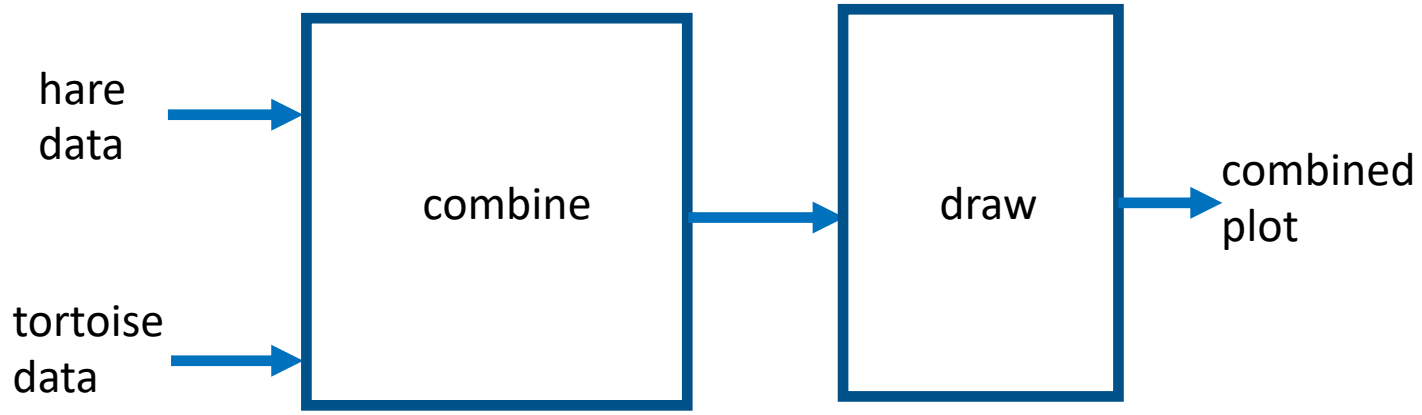



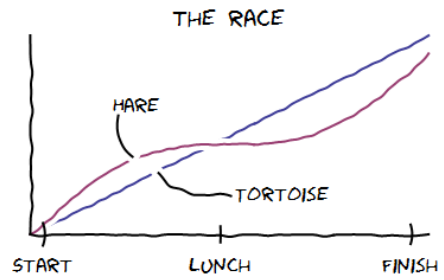
# Revisit toy pipeline: both plots by modularization

| racer | time | dist |
|---|---|---|
| hare | START | 0 |
| hare | 1 | 0.25 |
| | … | |
| hare | LUNCH | 0.5 |
| | … | |
| hare | FINISH | 0.9 |
| tort | START | 0 |
| tort | 1 | 0.1 |
| | … | |
| tort | LUNCH | 0.5 |
| | … | |
| tort | FINISH | 1 |



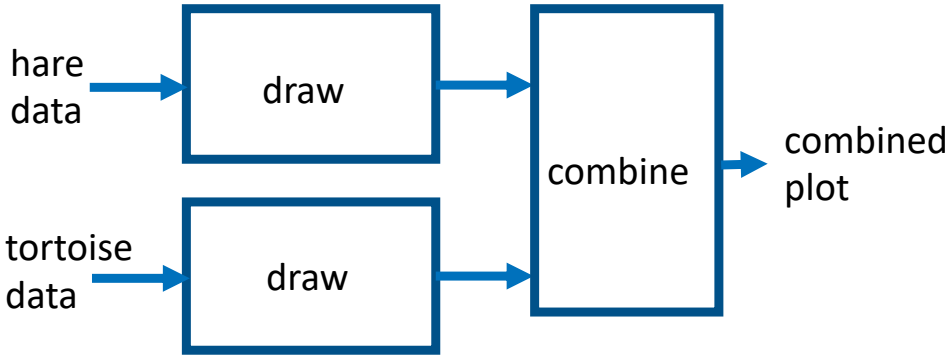



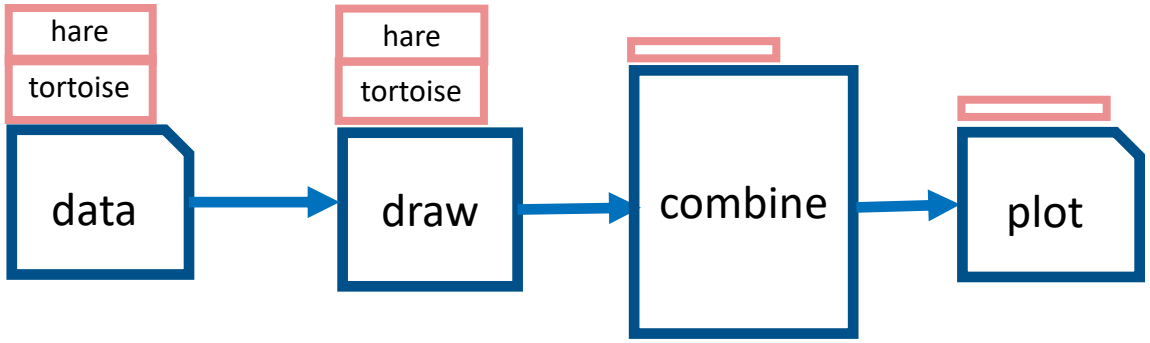



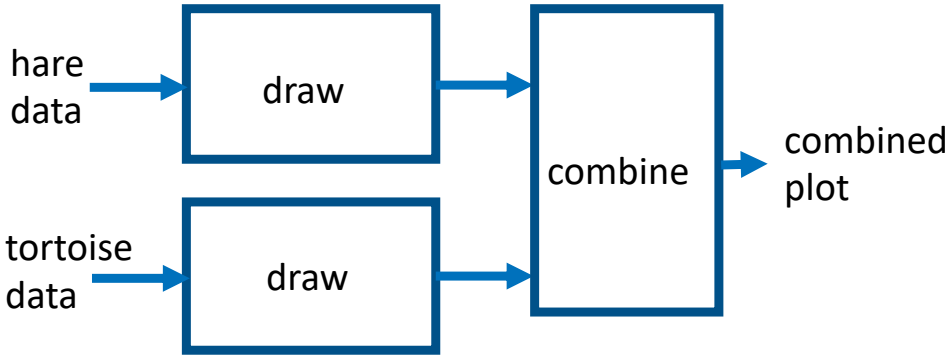
# Toy pipeline: modularization & parameterization

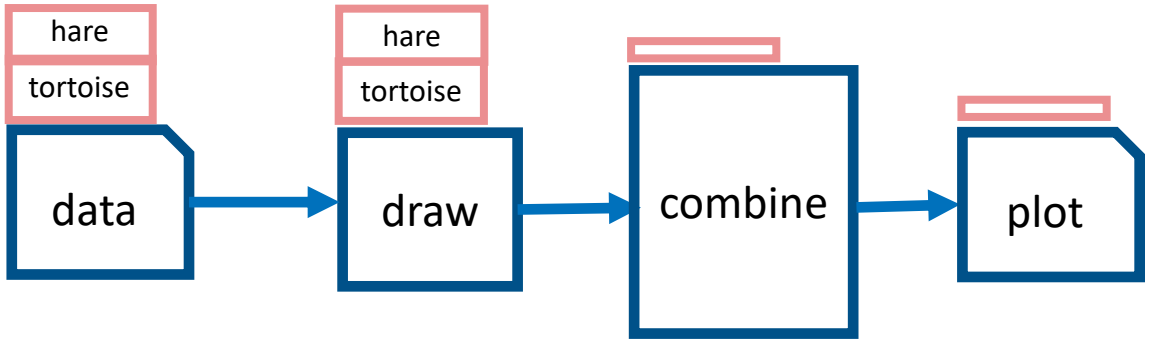
# Toy pipeline: modularization & parameterization

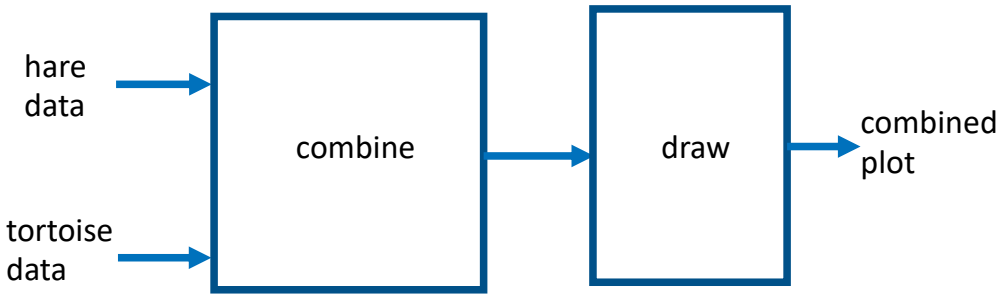


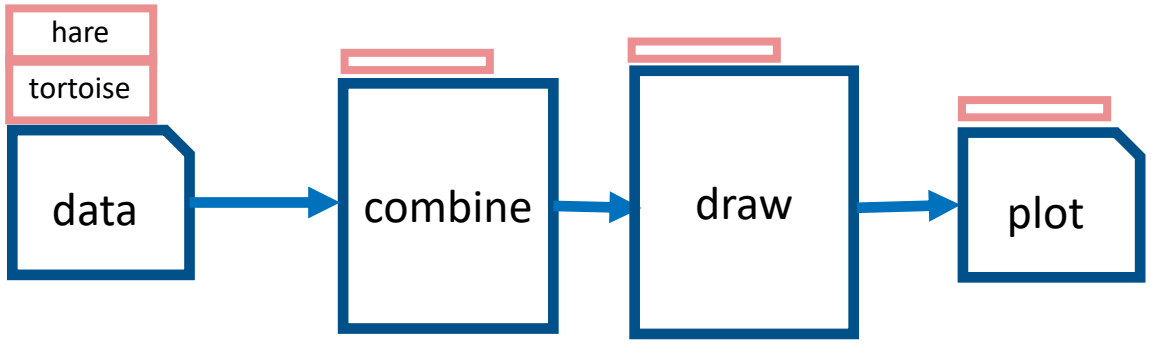



# Toy pipeline: modularization & parameterization

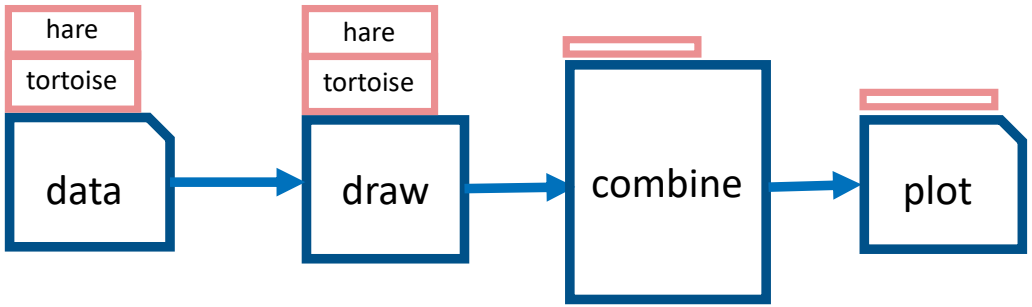


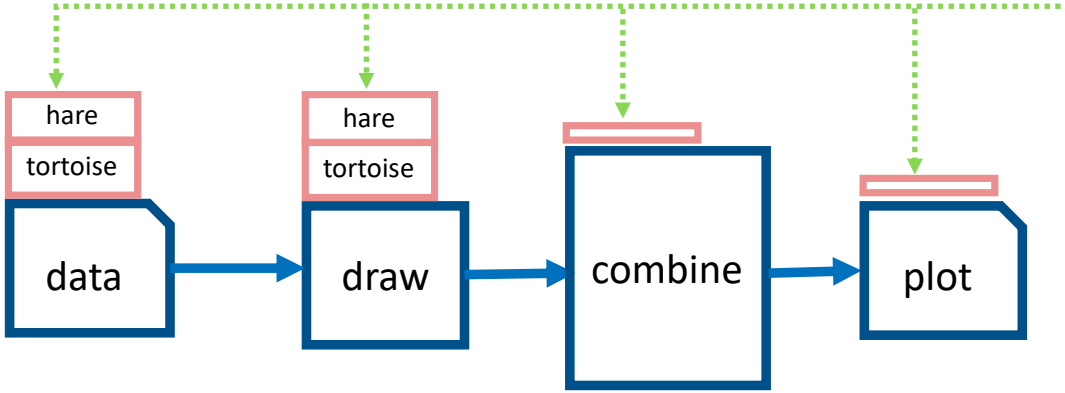



# Key ingredients of a JUDI pipeline

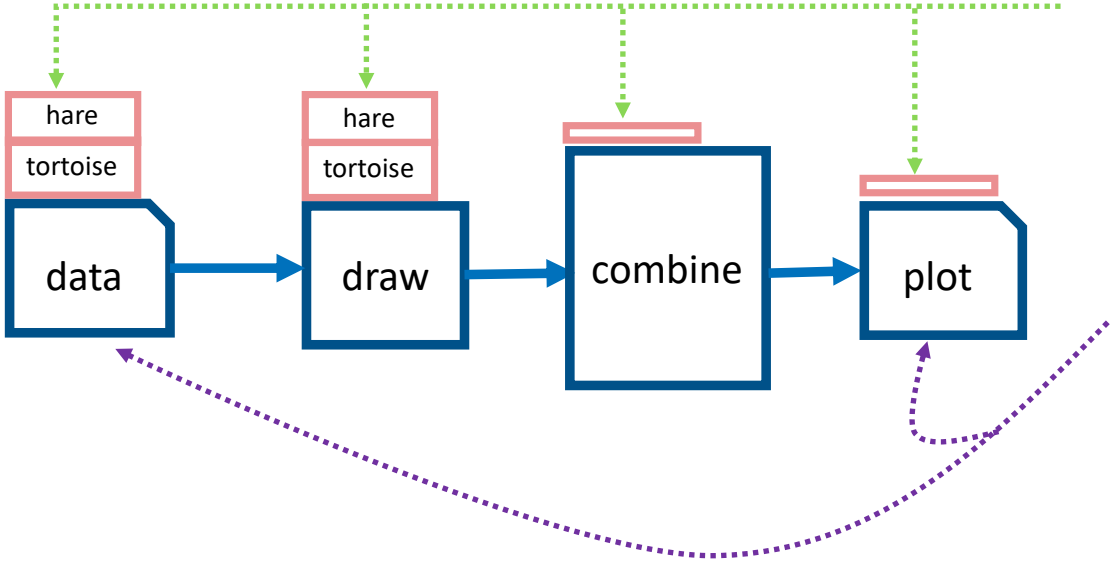


# Key ingredients of a JUDI pipeline



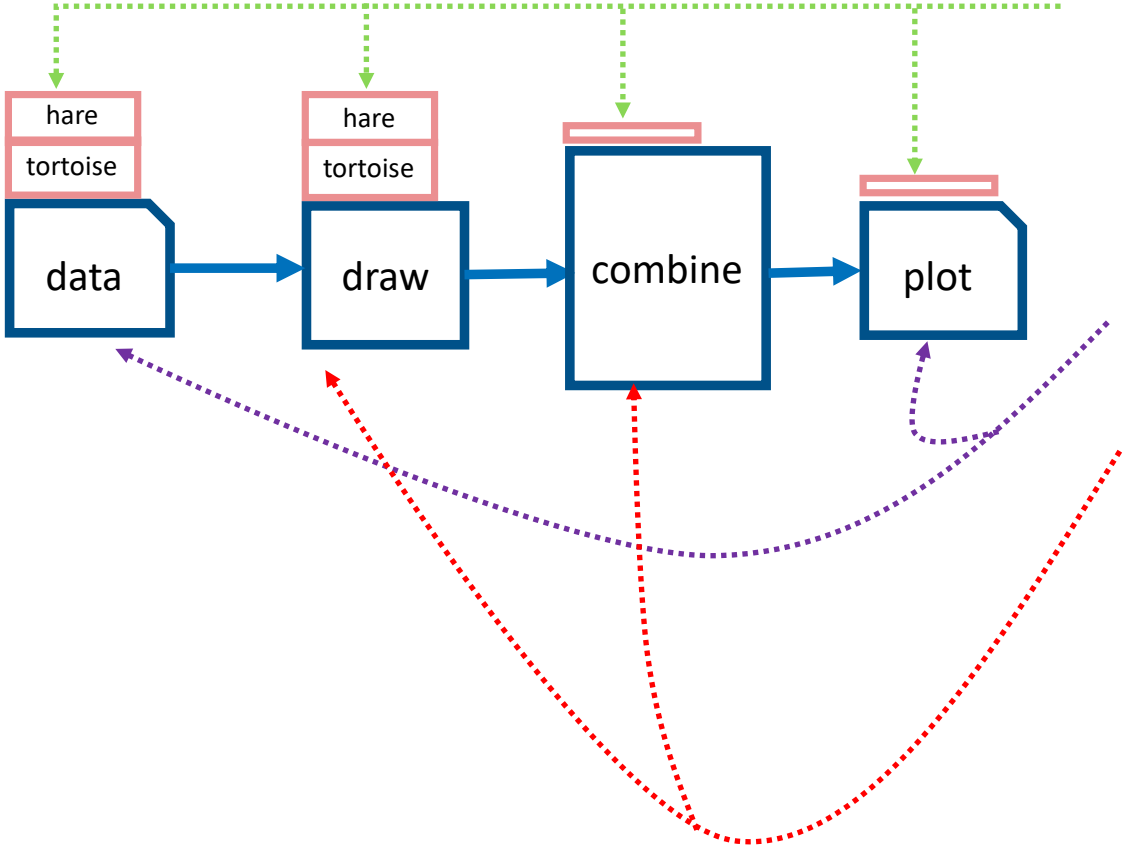

- Parameter database

# Key ingredients of a JUDI pipeline



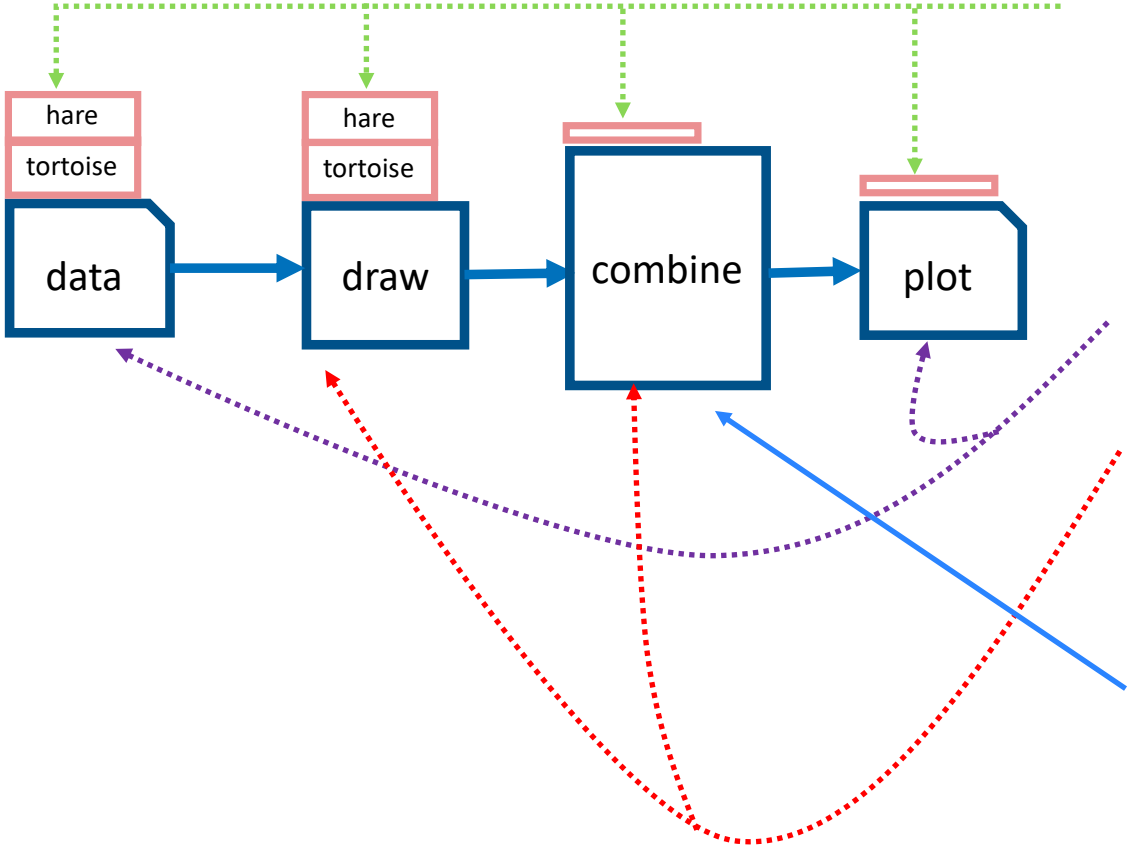

- Parameter database
- File

# Key ingredients of a JUDI pipeline



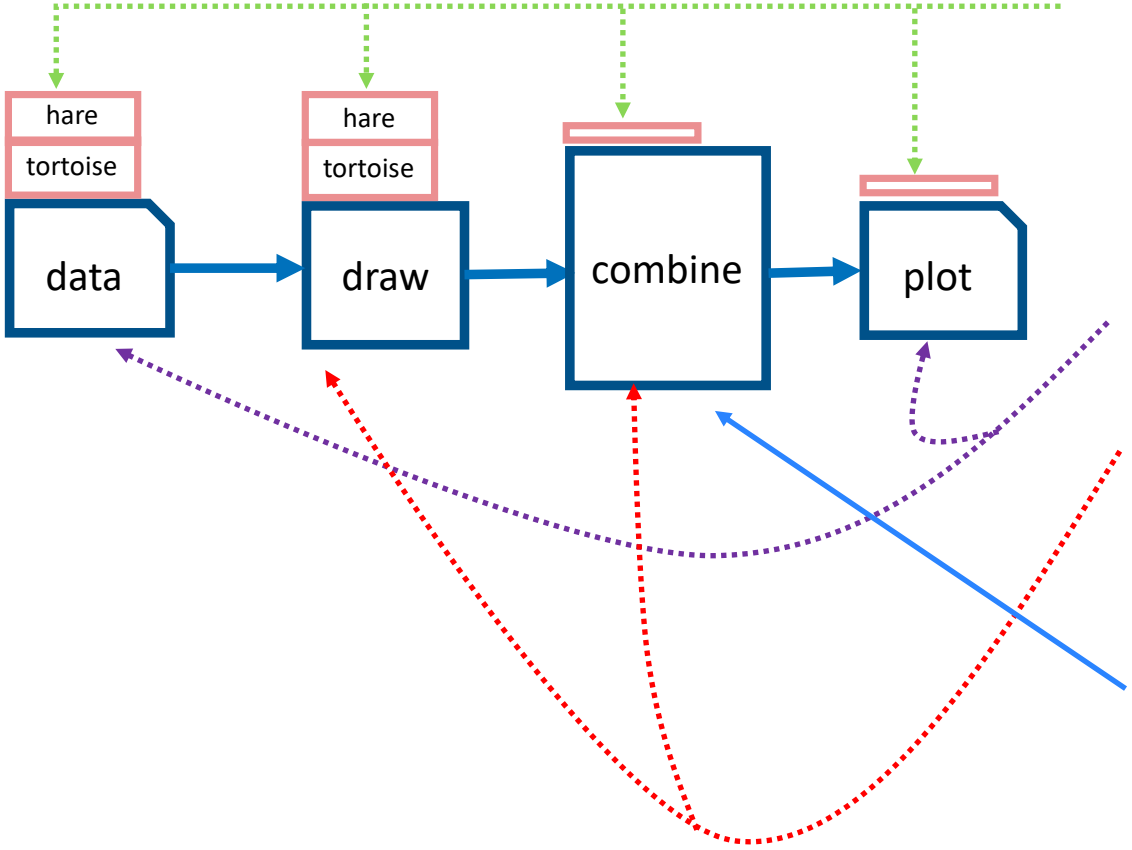

- Parameter database
- File
- Tasks

# Key ingredients of a JUDI pipeline



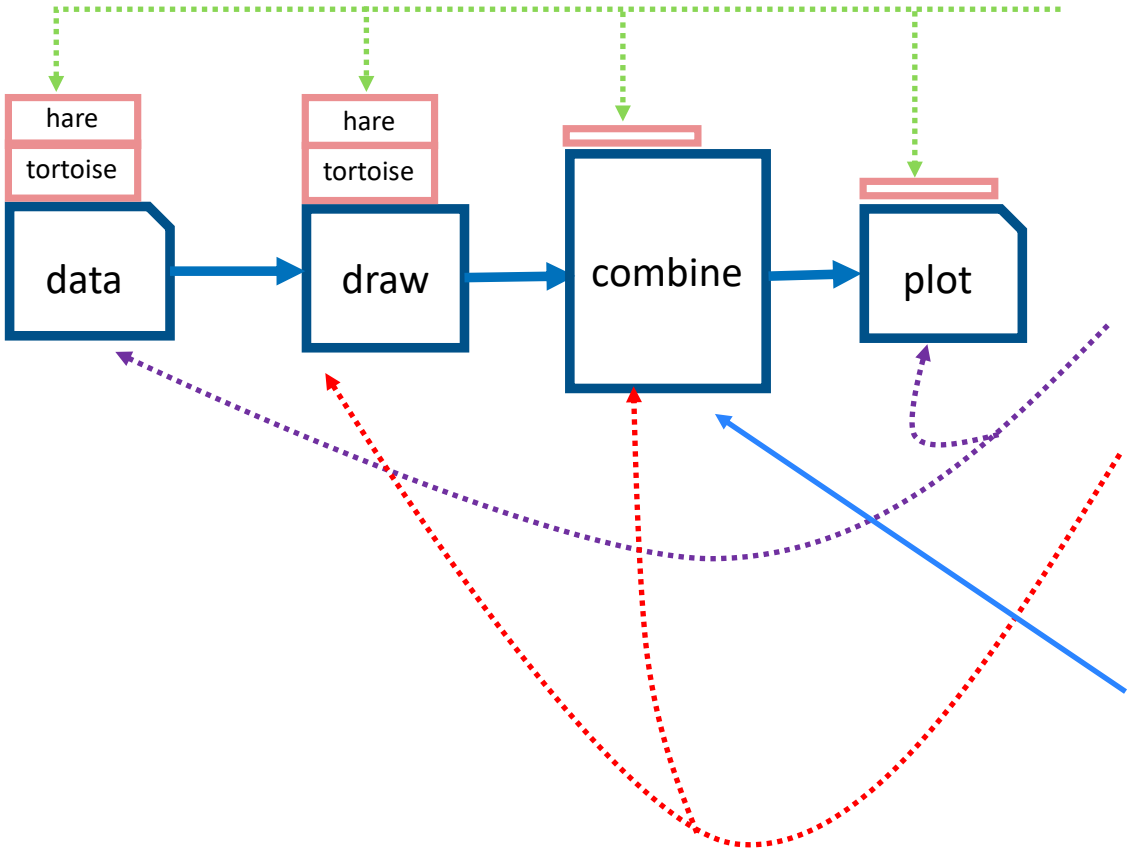

- Parameter database
- File
- Tasks
- Library of tasks for collation

# Key ingredients of a JUDI pipeline




- Parameter database
- File
- Tasks
- Library of tasks for collation
- Execution engine (**`doit`**)

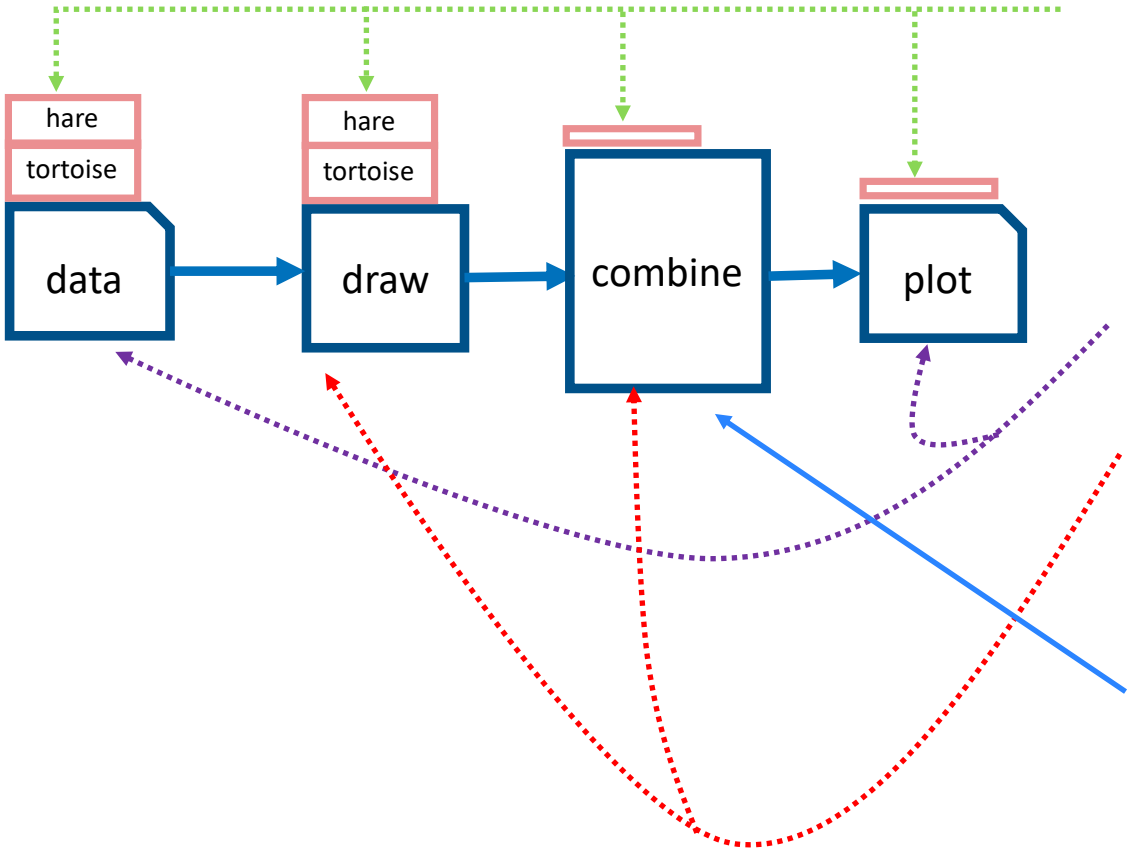
# Key ingredients of a JUDI pipeline




- Parameter database
- File
- Tasks
- Library of tasks for collation
- Execution engine (**`doit`**)

Provide a way to capture variability

# Key ingredients of a JUDI pipeline

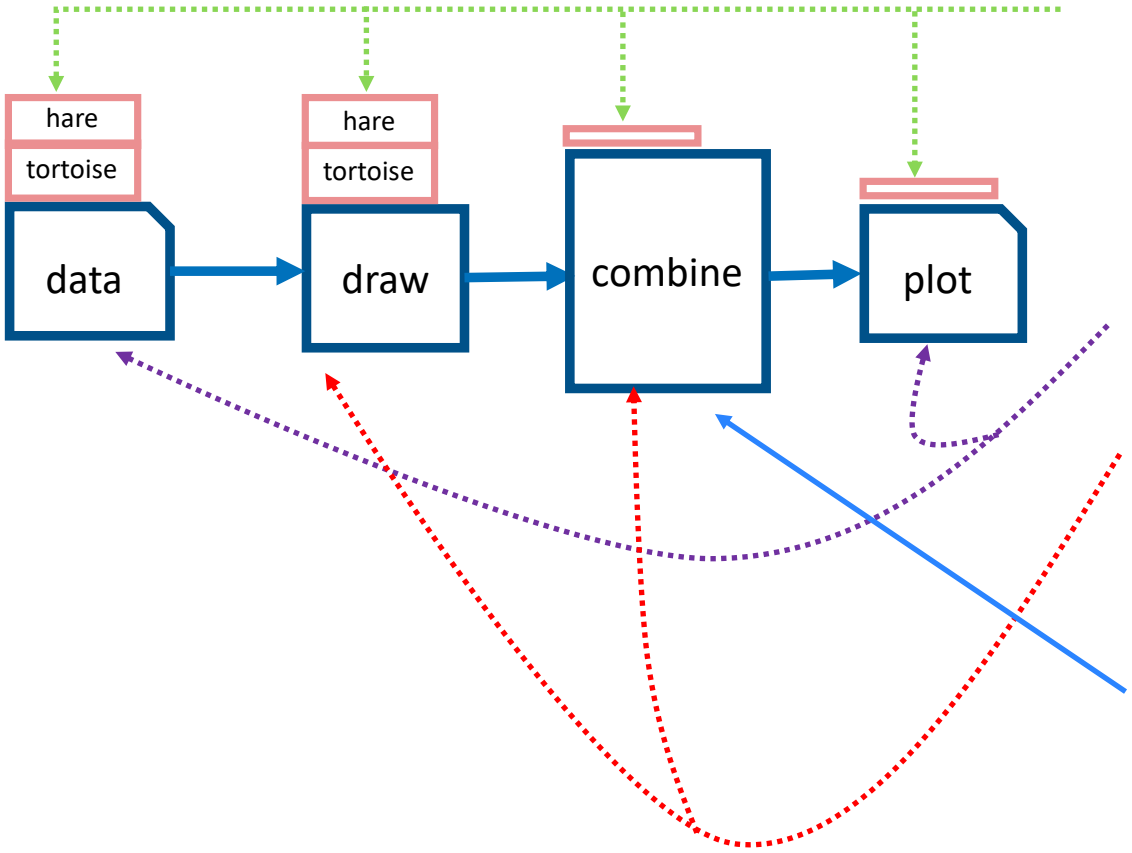



- Parameter database
- File
- Tasks
- Library of tasks for collation
- Execution engine (**`doit`**)

Provide a way to capture variability

Way to collate results

# Key ingredients of a JUDI pipeline




- Parameter database
- File
- Tasks
- Library of tasks for collation
- Execution engine (**`doit`**)

Provide a way to capture variability

Way to collate results

Optimal execution under different parameter values

# Real example pipeline

# Real example pipeline

*Align sample sequence reads to ref genome and compare coverage*

# Real example pipeline

*Align sample sequence reads to ref genome and compare coverage*

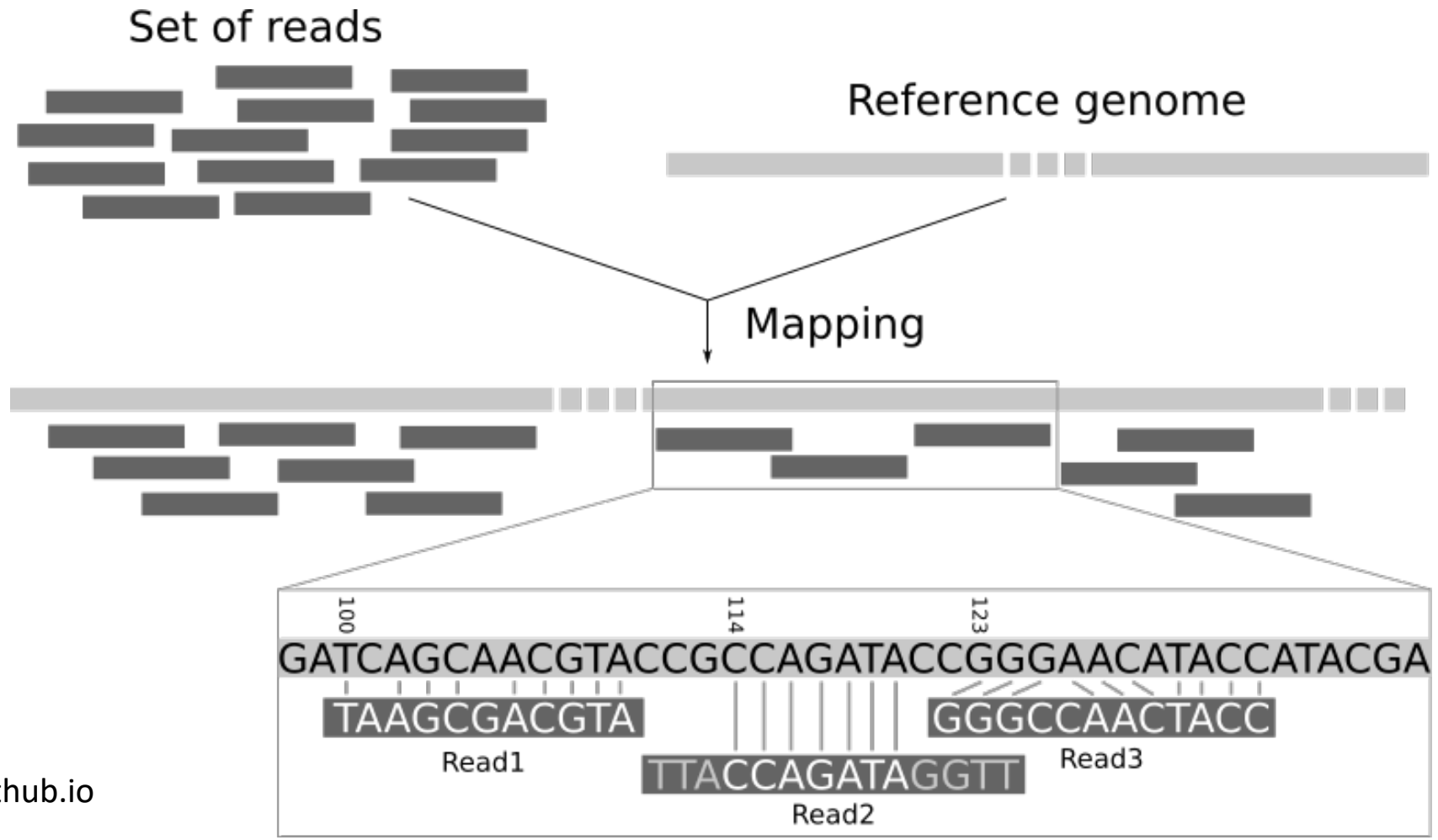



Image credit:
galaxyproject.github.io

# Real example pipeline

*Align sample sequence reads to ref genome and compare coverage*

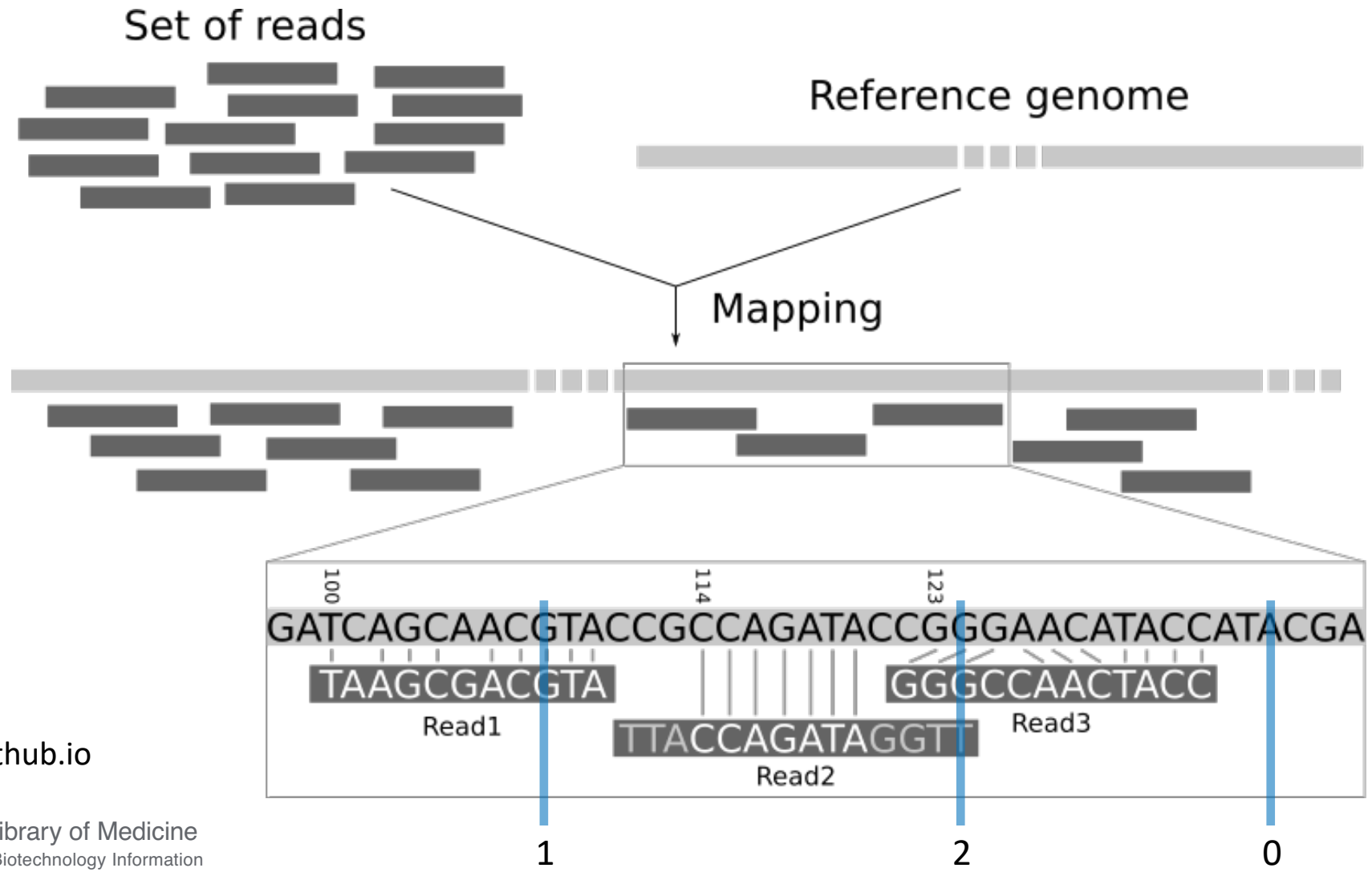



Image credit:
galaxyproject.github.io

# Input & output of example pipeline

```
@SRR038845.3 HWI-EAS038:6:1:0:1938 length=36
CAACGAGTTCACACCTTGGCCGACAGGCCCGGGTAA
+SRR038845.3 HWI-EAS038:6:1:0:1938 length=36
BA@7>B=>:>>7@7@>>9=BAA?;>52;>:9=8.=A
@SRR038845.41 HWI-EAS038:6:1:0:1474 length=36
CCAATGATTTTTTTCCGTGTTTCAGAATACGGTTAA
+SRR038845.41 HWI-EAS038:6:1:0:1474 length=36
BCCBA@BB@BBBBAB@B9B@=BABA@A:@693:@B=
@SRR038845.53 HWI-EAS038:6:1:1:360 length=36
GTTCAAAAAGAACTAAATTGTGTCAATAGAAAACTC
+SRR038845.53 HWI-EAS038:6:1:1:360 length=36
BBCBBBBBB@@BAB?BBBBCBC>BBBAA8>BBBAA@
```

...

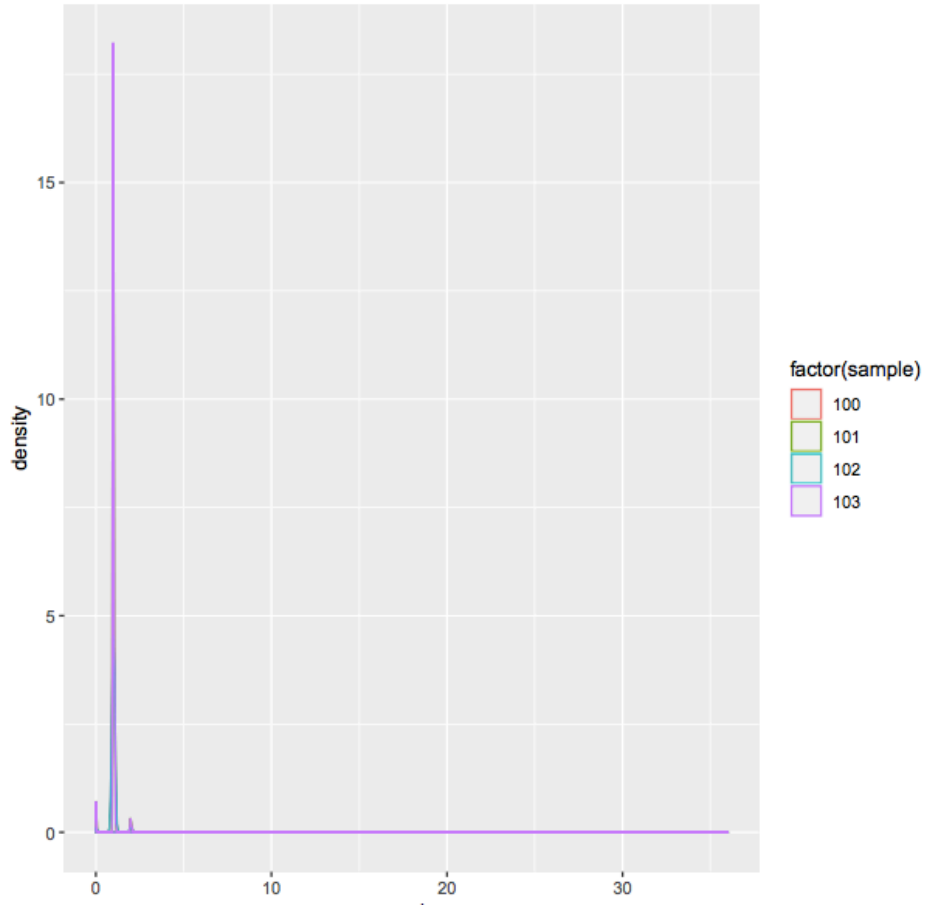
```
@SRR038845.3 HWI-EAS038:6:1:0:1938 length=36
CAACGAGTTCACACCTTGGCCGACAGGCCCGGGTAA
+SRR038845.3 HWI-EAS038:6:1:0:1938 length=36
BA@7>B=>:>>7@7@>>9=BAA?;>52;>:9=8.=A
@SRR038845.41 HWI-EAS038:6:1:0:1474 length=36
CCAATGATTTTTTTCCGTGTTTCAGAATACGGTTAA
+SRR038845.41 HWI-EAS038:6:1:0:1474 length=36
BCCBA@BB@BBBBAB@B9B@=BABA@A:@693:@B=
@SRR038845.53 HWI-EAS038:6:1:1:360 length=36
GTTCAAAAAGAACTAAATTGTGTCAATAGAAAACTC
+SRR038845.53 HWI-EAS038:6:1:1:360 length=36
BBCBBBBBB@@BAB?BBBBCBC>BBBAA8>BBBAA@
```

# Input & output of example pipeline

```
@SRR038845.3 HWI-EAS038:6:1:0:1938 length=36
CAACGAGTTCACACCTTGGCCGACAGGCCCGGGTAA
+SRR038845.3 HWI-EAS038:6:1:0:1938 length=36
BA@7>B=>:>>7@7@>>9=BAA?;>52;>:9=8.=A
@SRR038845.41 HWI-EAS038:6:1:0:1474 length=36
CCAATGATTTTTTTCCGTGTTTCAGAATACGGTTAA
+SRR038845.41 HWI-EAS038:6:1:0:1474 length=36
BCCBA@BB@BBBBAB@B9B@=BABA@A:@693:@B=
@SRR038845.53 HWI-EAS038:6:1:1:360 length=36
GTTCAAAAAGAACTAAATTGTGTCAATAGAAAACTC
+SRR038845.53 HWI-EAS038:6:1:1:360 length=36
BBCBBBBBB@@BAB?BBBBCBC>BBBAA8>BBBAA@
```

...


```
@SRR038845.3 HWI-EAS038:6:1:0:1938 length=36
CAACGAGTTCACACCTTGGCCGACAGGCCCGGGTAA
+SRR038845.3 HWI-EAS038:6:1:0:1938 length=36
BA@7>B=>:>>7@7@>>9=BAA?;>52;>:9=8.=A
@SRR038845.41 HWI-EAS038:6:1:0:1474 length=36
CCAATGATTTTTTTCCGTGTTTCAGAATACGGTTAA
+SRR038845.41 HWI-EAS038:6:1:0:1474 length=36
BCCBA@BB@BBBBAB@B9B@=BABA@A:@693:@B=
@SRR038845.53 HWI-EAS038:6:1:1:360 length=36
GTTCAAAAAGAACTAAATTGTGTCAATAGAAAACTC
+SRR038845.53 HWI-EAS038:6:1:1:360 length=36
BBCBBBBBB@@BAB?BBBBCBC>BBBAA8>BBBAA@
```

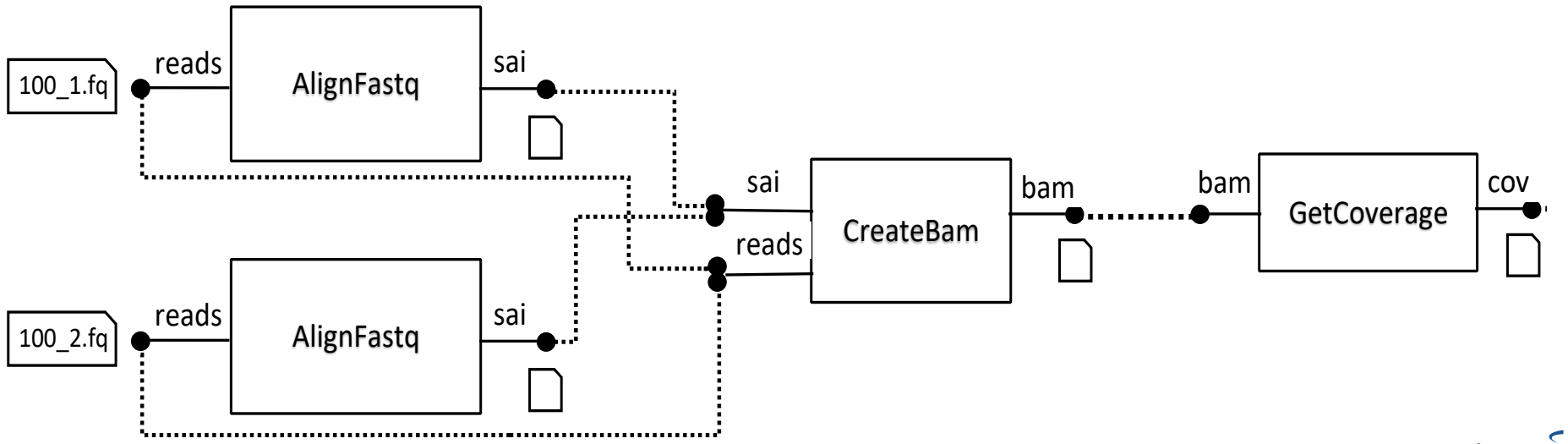
# Stages of real example pipeline

# Stages of real example pipeline

| Serial No | Sample | Group |
|---|---|---|
| 1 | 100 | 1 |
| 2 | 100 | 2 |
| 3 | 101 | 1 |
| 4 | 101 | 2 |
| 5 | 102 | 1 |
| 6 | 102 | 2 |
| 7 | 103 | 1 |
| 8 | 103 | 2 |

# Stages of real example pipeline

| Serial No | Sample | Group |
|---|---|---|
| 1 | 100 | 1 |
| 2 | 100 | 2 |
| 3 | 101 | 1 |
| 4 | 101 | 2 |
| 5 | 102 | 1 |
| 6 | 102 | 2 |
| 7 | 103 | 1 |
| 8 | 103 | 2 |

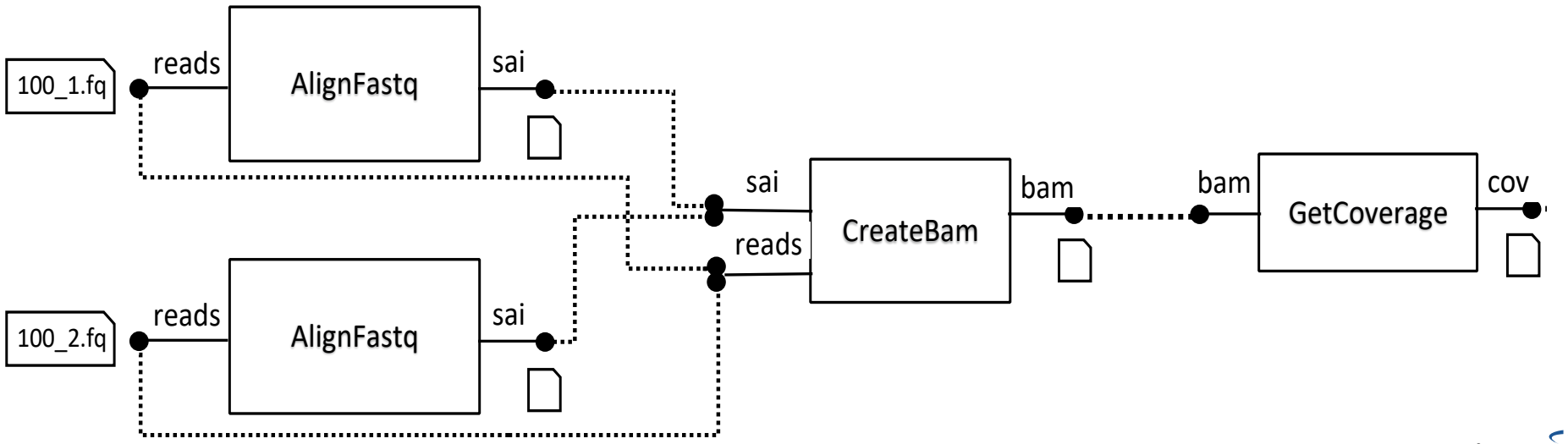


# Stages of real example pipeline

| Serial No | Sample | Group |
|---|---|---|
| 1 | 100 | 1 |
| 2 | 100 | 2 |
| 3 | 101 | 1 |
| 4 | 101 | 2 |
| 5 | 102 | 1 |
| 6 | 102 | 2 |
| 7 | 103 | 1 |
| 8 | 103 | 2 |

- Align each fastq file to a ref genome and create temp aln file

bwa tool



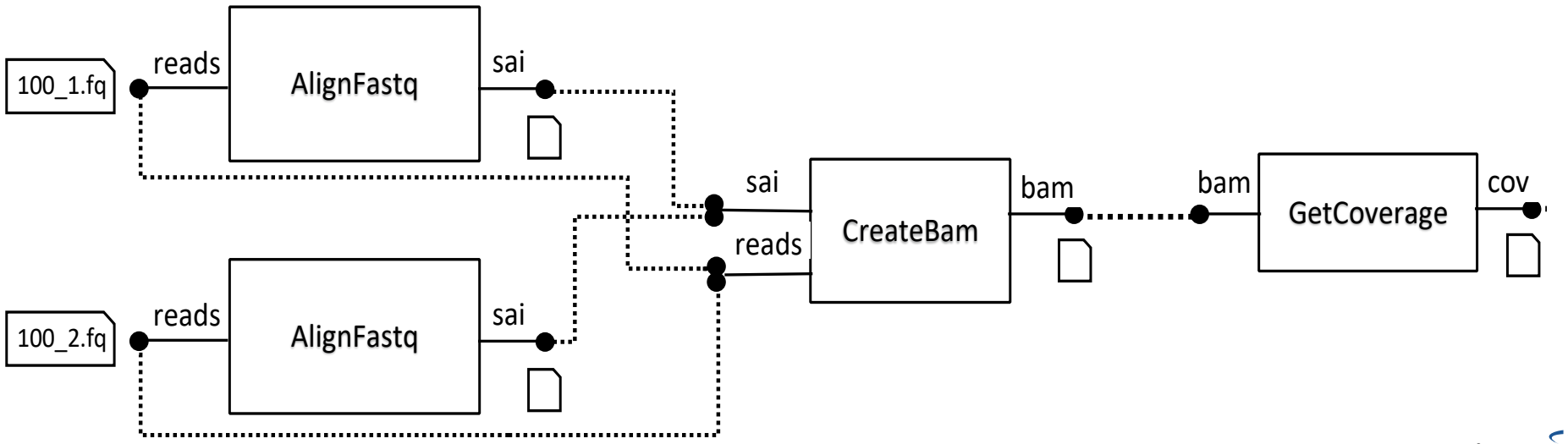
# Stages of real example pipeline

| Serial No | Sample | Group |
|---|---|---|
| 1 | 100 | 1 |
| 2 | 100 | 2 |
| 3 | 101 | 1 |
| 4 | 101 | 2 |
| 5 | 102 | 1 |
| 6 | 102 | 2 |
| 7 | 103 | 1 |
| 8 | 103 | 2 |

- Align each fastq file to a ref genome and create temp aln file

bwa tool

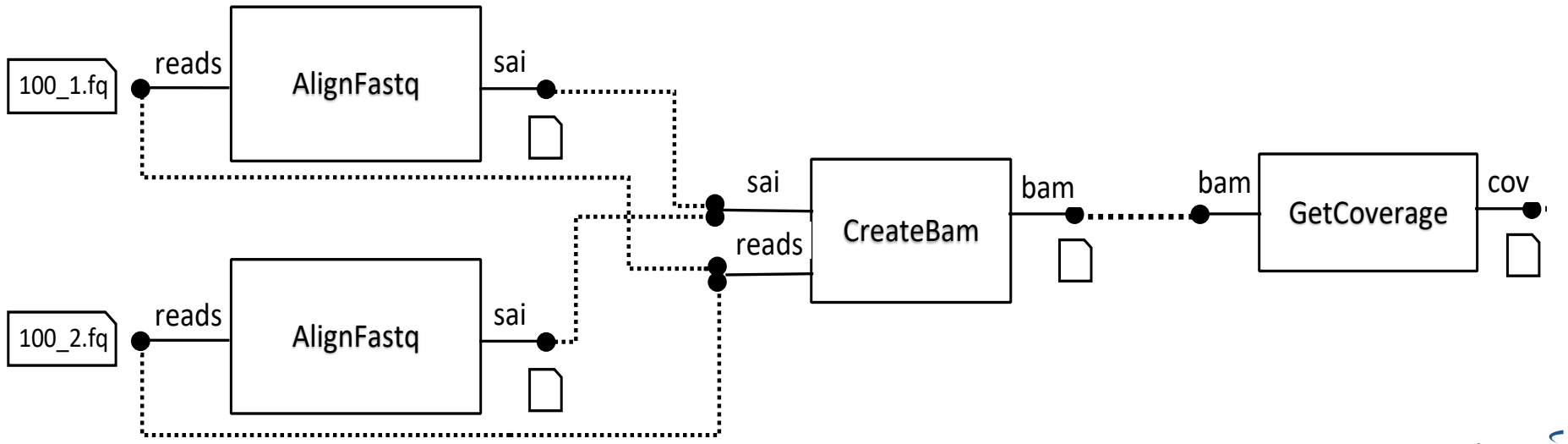
- Combine two temp file for each sample to a single bam file



# Stages of real example pipeline

| Serial No | Sample | Group |
|---|---|---|
| 1 | 100 | 1 |
| 2 | 100 | 2 |
| 3 | 101 | 1 |
| 4 | 101 | 2 |
| 5 | 102 | 1 |
| 6 | 102 | 2 |
| 7 | 103 | 1 |
| 8 | 103 | 2 |

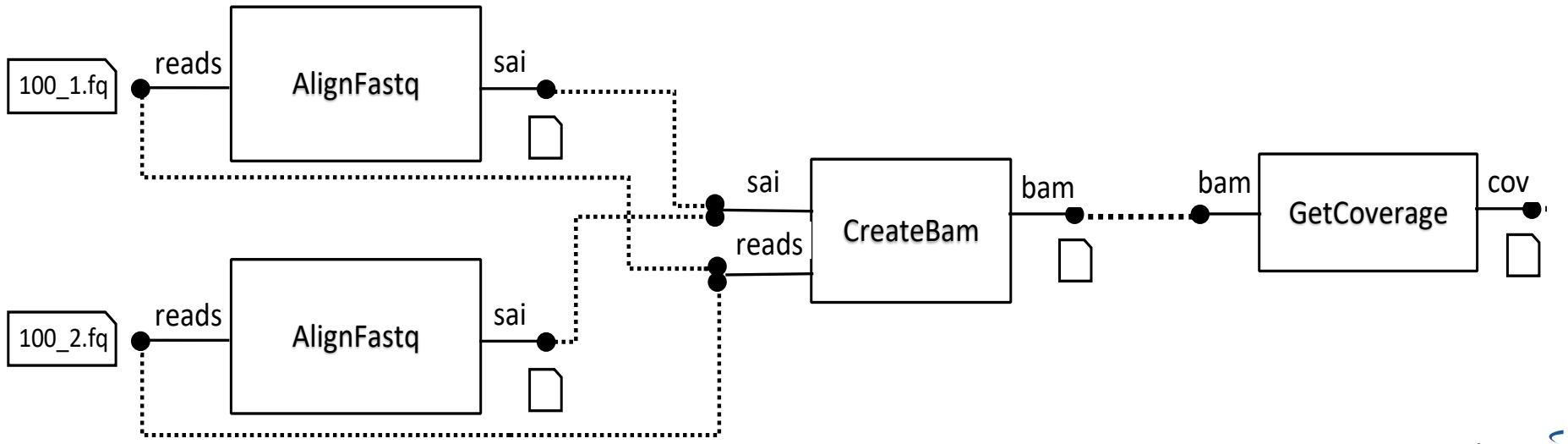
- Align each fastq file to a ref genome and create temp aln file bwa tool
- Combine two temp file for each sample to a single bam file
- Get coverage info from each sample BAM to a csv file samtools



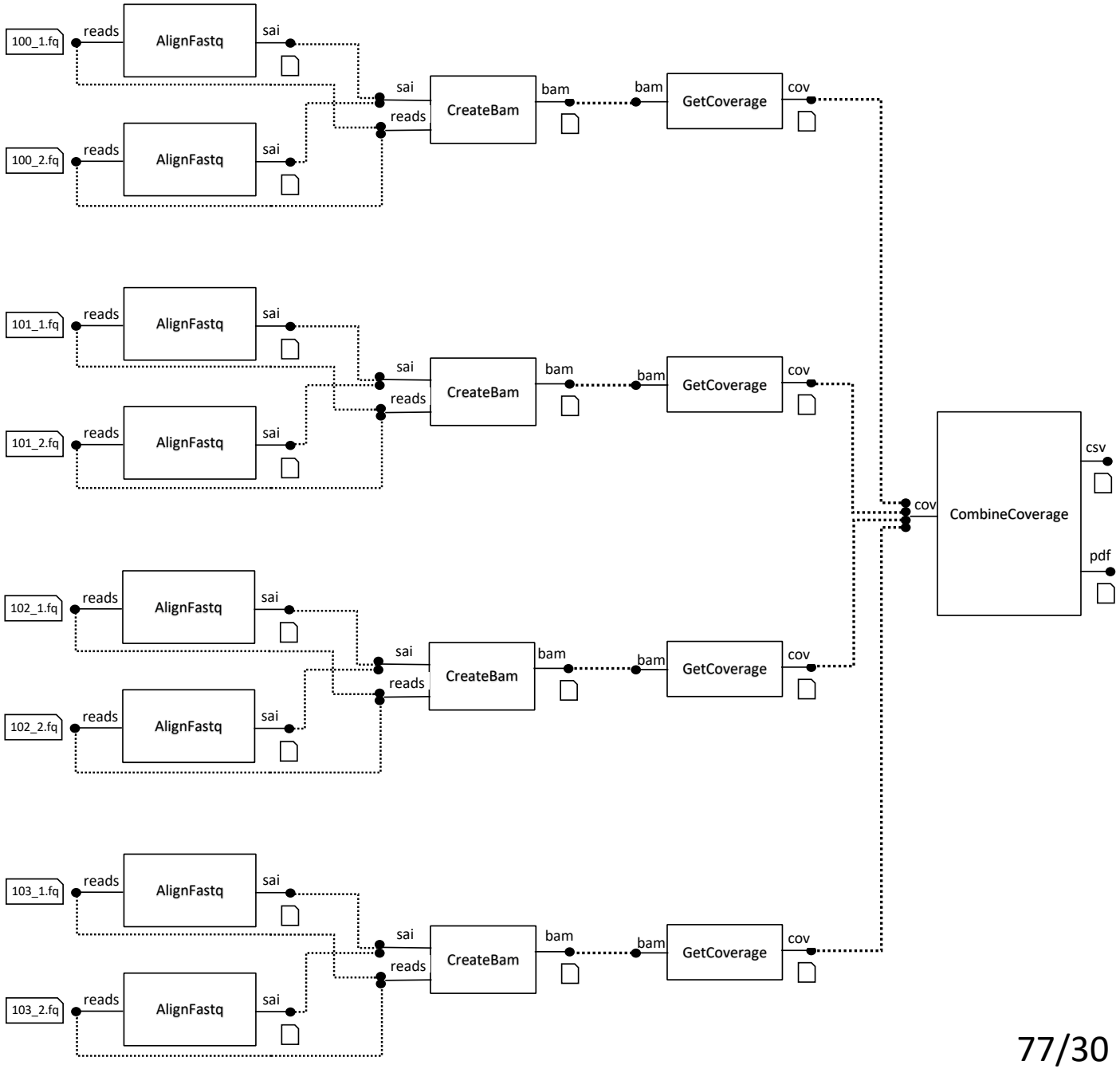
# Stages of real example pipeline

| Serial No | Sample | Group |
|---|---|---|
| 1 | 100 | 1 |
| 2 | 100 | 2 |
| 3 | 101 | 1 |
| 4 | 101 | 2 |
| 5 | 102 | 1 |
| 6 | 102 | 2 |
| 7 | 103 | 1 |
| 8 | 103 | 2 |

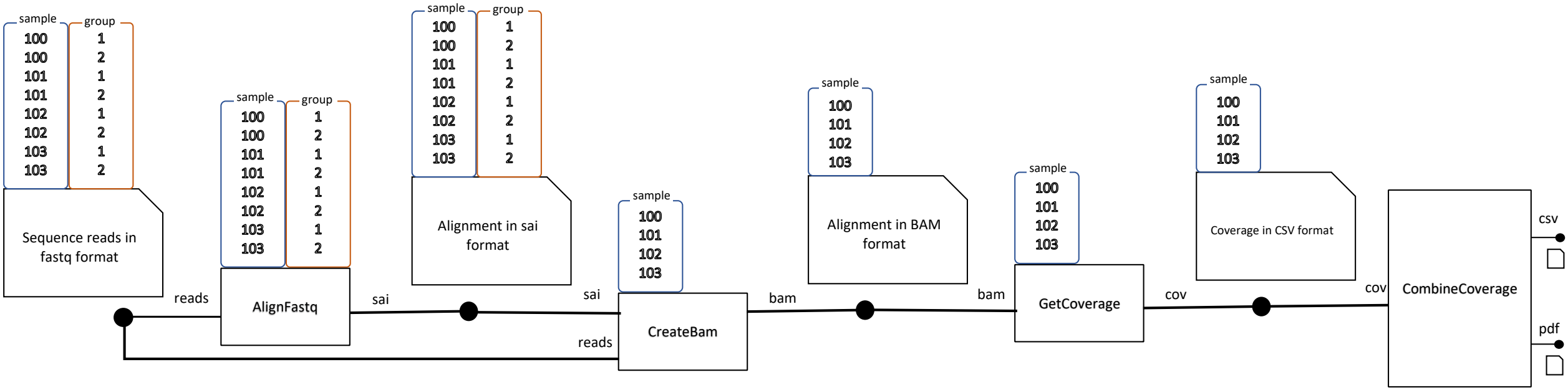
- Align each fastq file to a ref genome and create temp aln file bwa tool
- Combine two temp file for each sample to a single bam file
- Get coverage info from each sample BAM to a csv file samtools
- Combine coverage info from all samples and plot histogram



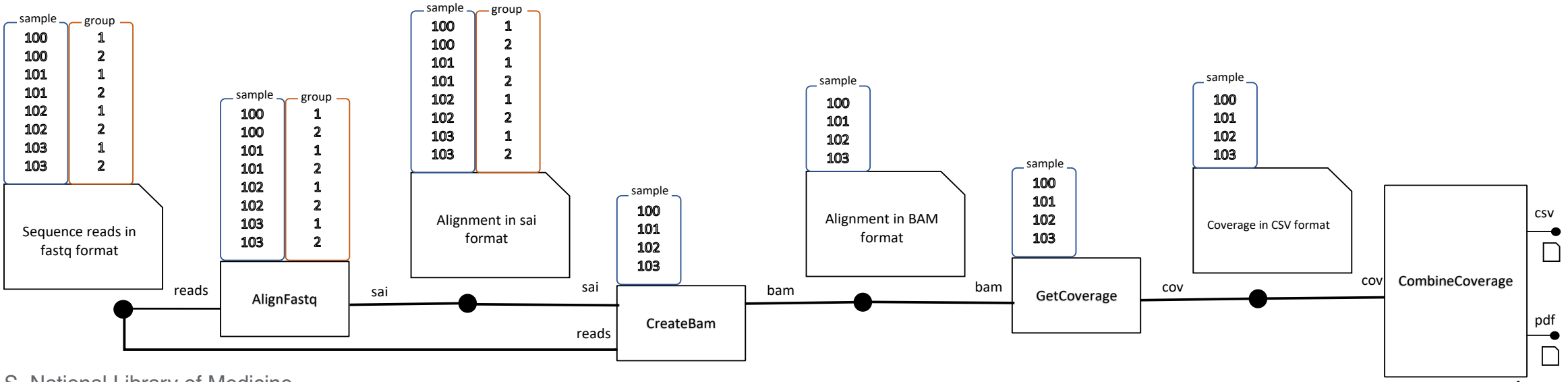
# Real example complete pipeline



# Real pipeline: JUDI view

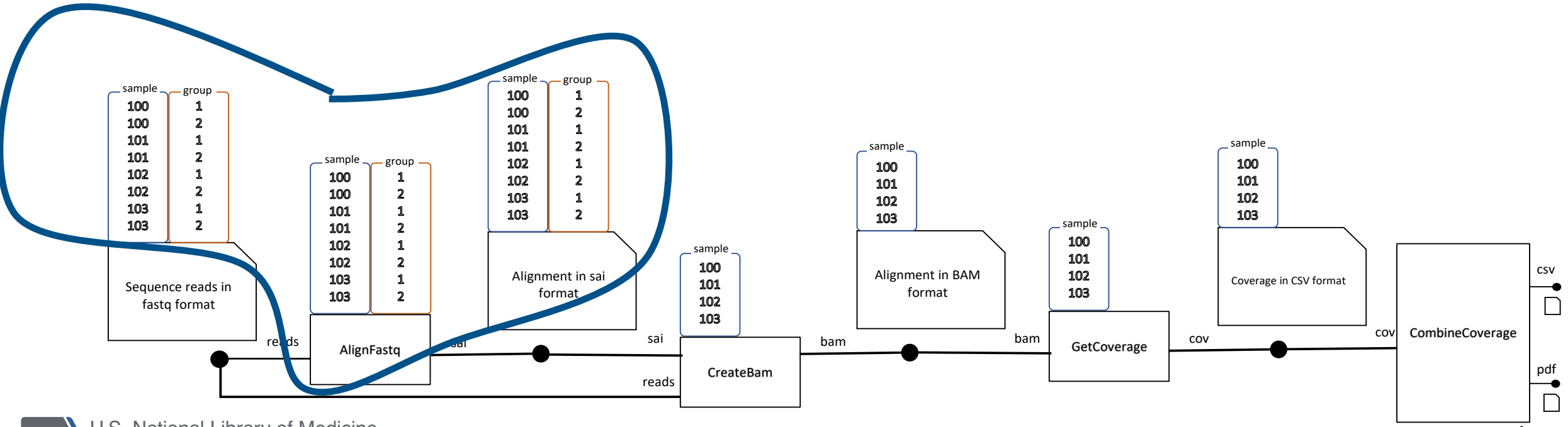


# Define ParamDb in JUDI



# Define ParamDb in JUDI

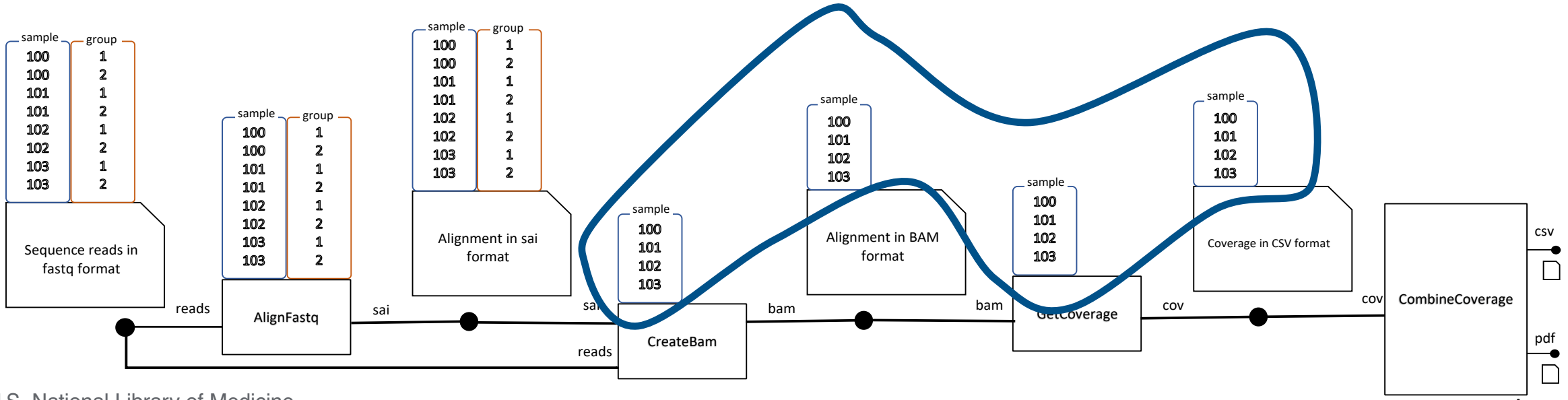
```
pdb_sample_grp = ParamDb('sample_grp')
pdb_sample_grp.add_param('100 101 102 103'.split(), 'sample')
pdb_sample_grp.add_param('1 2'.split(), 'group')
```



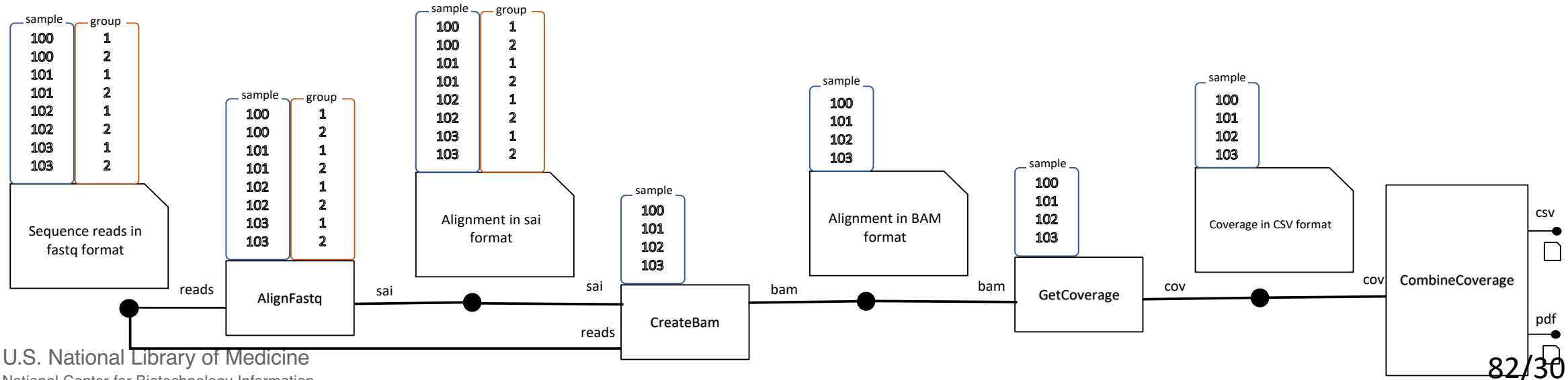
# Define ParamDb in JUDI

```
pdb_sample_grp = ParamDb('sample_grp')
pdb_sample_grp.add_param('100 101 102 103'.split(), 'sample')
pdb_sample_grp.add_param('1 2'.split(), 'group')
```

```
pdb_sample = ParamDb('sample')
pdb_sample.add_param('100 101 102 103'.split(), 'sample')
```

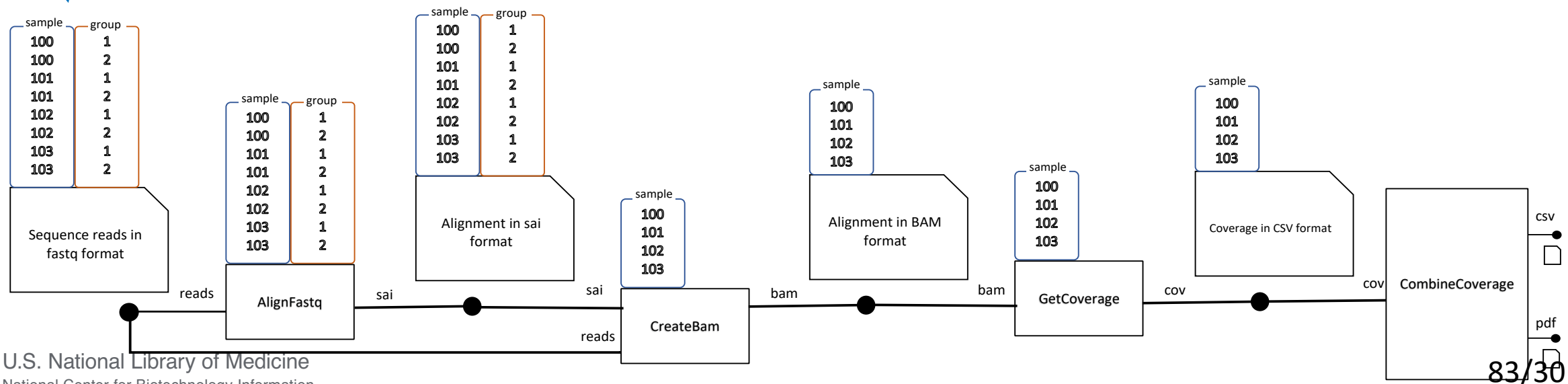


# Define File in JUDI



# Define File in JUDI

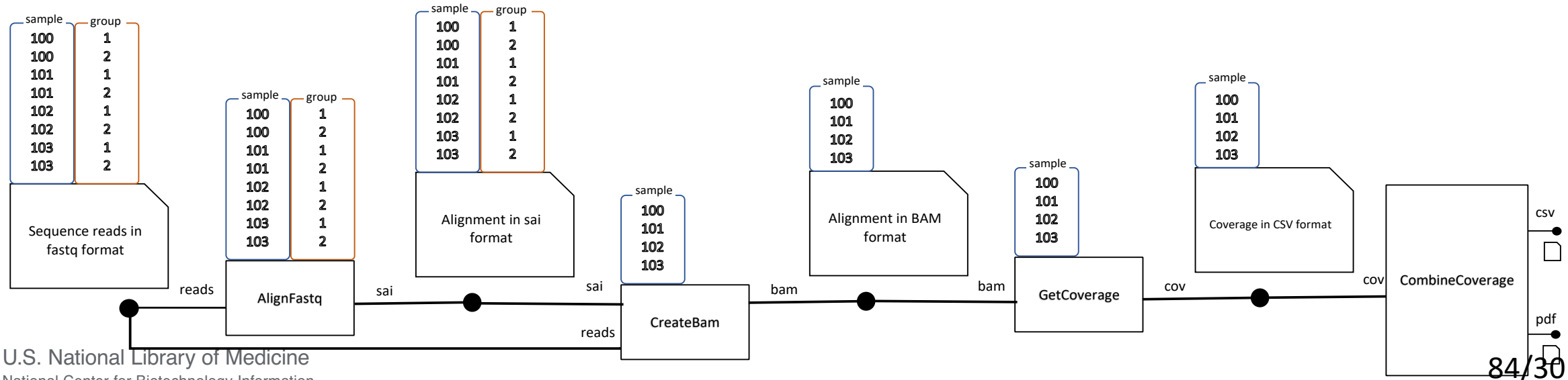
```
path_gen = lambda x: '{}_{}.fq'.format(x['sample'],x['group'])
f_reads = File('seq', param = pdb_sample_grp, root = '.', path = path_gen)
```



# Define File in JUDI

```
path_gen = lambda x: '{}_{}.fq'.format(x['sample'],x['group'])
f_reads = File('seq', param = pdb_sample_grp, root = '.', path = path_gen)
```

```
f_aln = File('aln.sai', param = pdb_sample_grp)
```

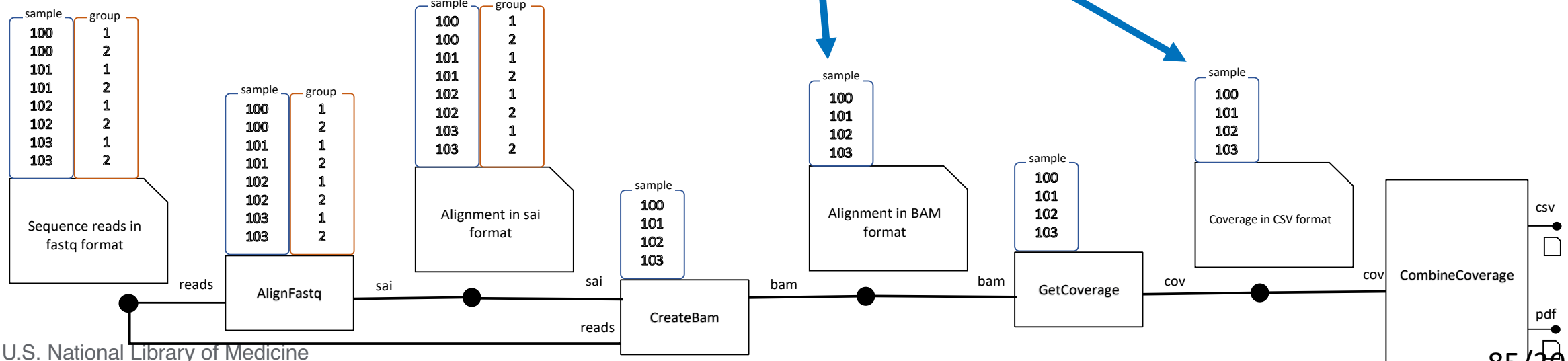


# Define File in JUDI

```
path_gen = lambda x: '{}_{}.fq'.format(x['sample'],x['group'])
f_reads = File('seq', param = pdb_sample_grp, root = '.', path = path_gen)
```

```
f_aln = File('aln.sai', param = pdb_sample_grp)
```

```
f_bam = File('aln.bam', param = pdb_sample)
f_cov = File('cov.csv', param = pdb_sample)
```



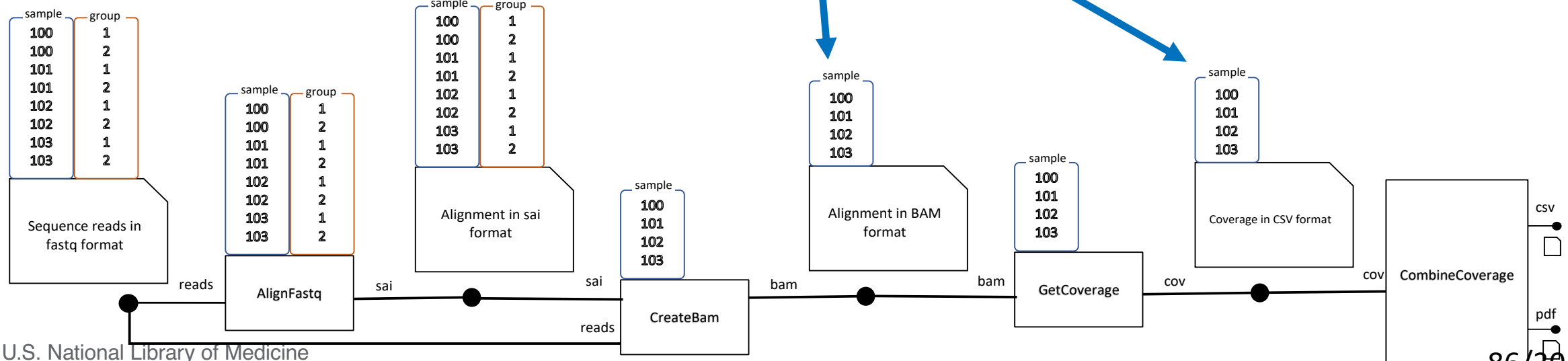
# Define File in JUDI

```
path_gen = lambda x: '{}_{}.fq'.format(x['sample'],x['group'])
f_reads = File('seq', param = pdb_sample_grp, root = '.', path = path_gen)
```

```
f_aln = File('aln.sai', param = pdb_sample_grp)
```

```
f_bam = File('aln.bam', param = pdb_sample)
f_cov = File('cov.csv', param = pdb_sample)
```

```
f_cov_combined = File('combined.csv')
f_plt_combined = File('plot.pdf', root = '.')
```



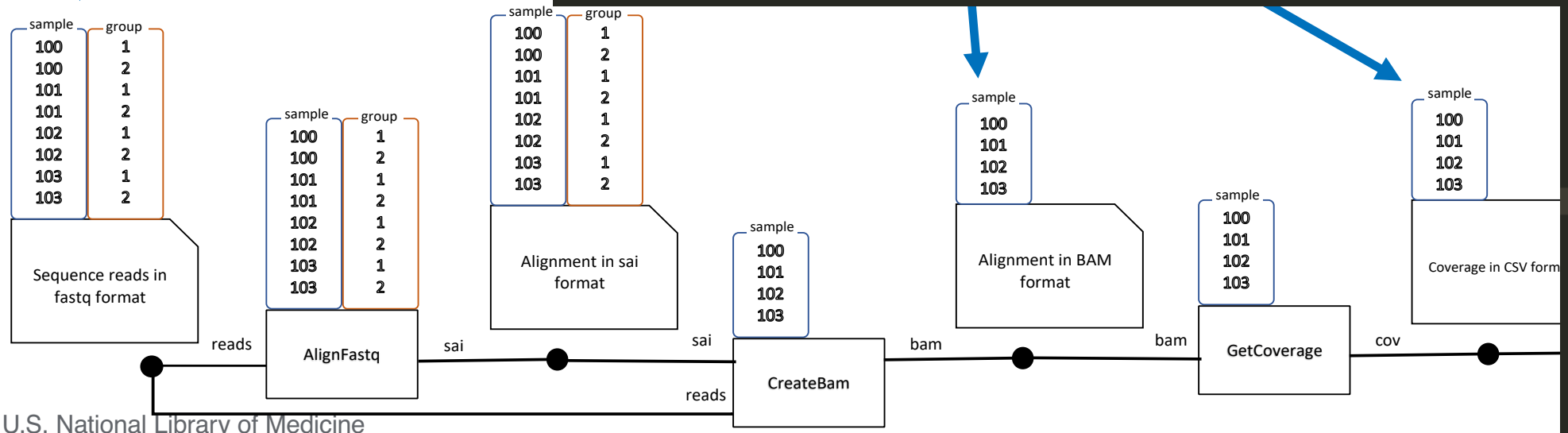
# Define File in JUDI

```
path_gen = lambda x: '{}_{}.fq'.format(x['sample'],x['group'])
f_reads = File('seq', param = pdb_sample_grp, root = '.', path = path_gen)
```

```
f_aln = File('aln.sai', param = pdb_sample_grp)
```

```
f_bam = File('aln.bam', param = pdb_sample)
f_cov = File('cov.csv', param = pdb_sample)
```

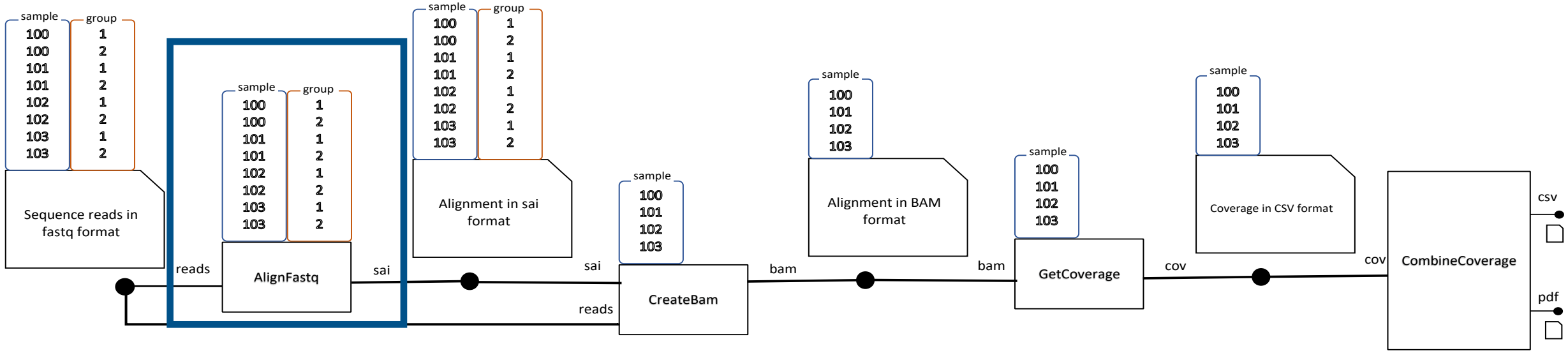
```
f_cov_combined = File('combined.csv')
f_plt_combined = File('plot.pdf', root = '.')
```




```
.
|— 100_1.fq
|— 100_2.fq
|— 101_1.fq
|— 101_2.fq
|— 102_1.fq
|— 102_2.fq
|— 103_1.fq
|— 103_2.fq
|
|— judi_files
|   |— combined.csv
|   |— group~1,sample~100
|   |   `— aln.sai
|   |— group~1,sample~101
|   |   `— aln.sai
|   |— group~1,sample~102
|   |   `— aln.sai
|   |— group~1,sample~103
|   |   `— aln.sai
|   |— group~2,sample~100
|   |   `— aln.sai
|   |— group~2,sample~101
|   |   `— aln.sai
|   |— group~2,sample~102
|   |   `— aln.sai
|   |— group~2,sample~103
|   |   `— aln.sai
|   |— sample~100
|   |   |— aln.bam
|   |   `— cov.csv
|   |— sample~101
|   |   |— aln.bam
|   |   `— cov.csv
|   |— sample~102
|   |   |— aln.bam
|   |   `— cov.csv
|   `— sample~103
|       |— aln.bam
|       `— cov.csv
|— combined.csv
`— plot.pdf
```

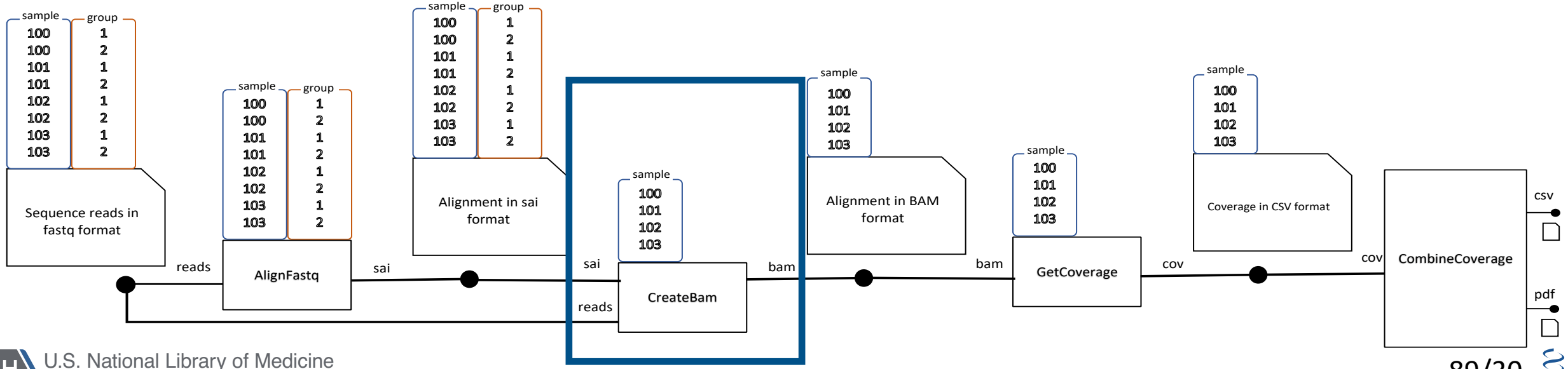
# AlignFastQ Task in JUDI

```
class AlignFastq(Task):
    param = pdb_sample_grp
    inputs = {'reads': f_reads}
    targets = {'sai': f_aln}
    actions = [('bwa aln {} {} > {}', [REF,'$reads','$sai'])]
```



# CreateBam Task in JUDI

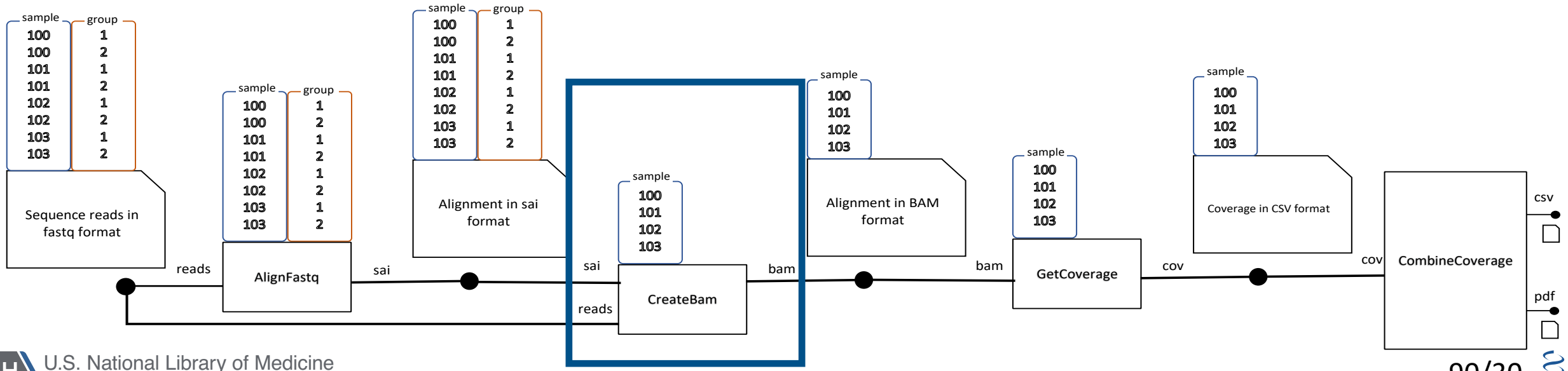
```
class CreateBam(Task):
    param = pdb_sample
    inputs = {'reads': f_reads, 'sai': f_aln}
    targets = {'bam': f_bam}
    actions = [('bwa sampe {} {} {} | samtools view -Sbh - '
                '| samtools sort - > {}', [REF,'$sai','$reads','$bam'])]
```



# CreateBam Task in JUDI

```
class CreateBam(Task):
    param = pdb_sample
    inputs = {'reads': f_reads, 'sai': f_aln}
    targets = {'bam': f_bam}
    actions = [('bwa sampe {} {} {} | samtools view -Sbh - '
                '| samtools sort - > {}', [REF,'$sai','$reads','$bam'])]
```

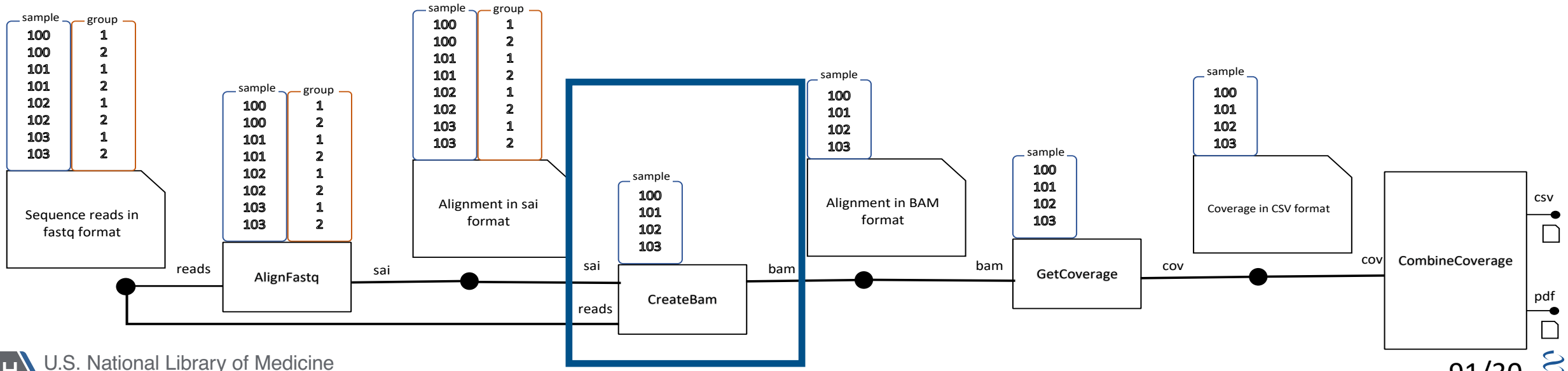
```
bwa sampe ref.fa  aln_sa1.sai aln_sa2.sai   read1.fq read2.fq  > aln-pe.sam
```



# CreateBam Task in JUDI

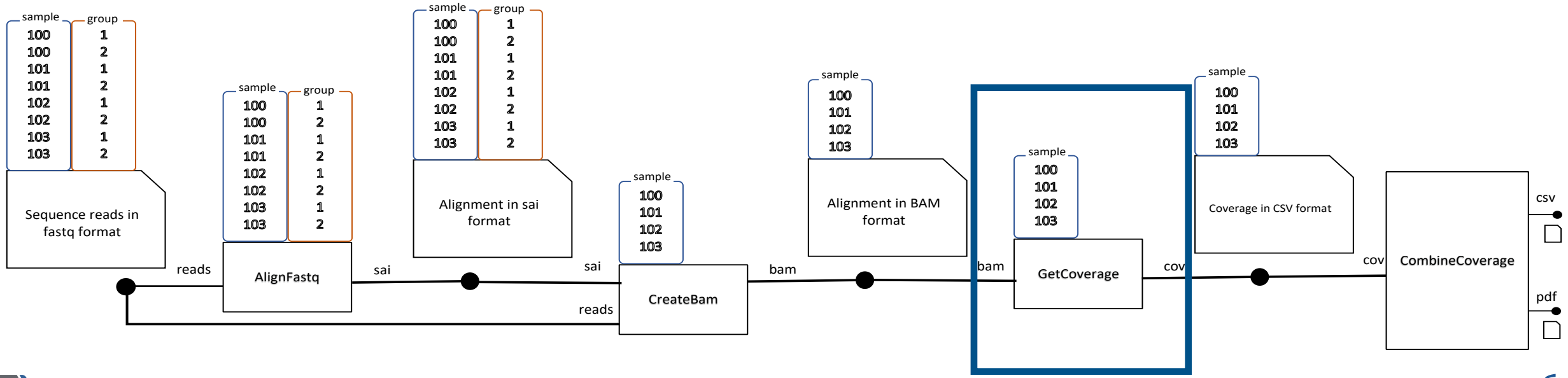
```
class CreateBam(Task):
    param = pdb_sample
    inputs = {'reads': f_reads, 'sai': f_aln}
    targets = {'bam': f_bam}
    actions = [('bwa sampe {} {} {} | samtools view -Sbh - '
                '| samtools sort - > {}', [REF,'$sai','$reads','$bam'])]
```

```
bwa sampe ref.fa {aln_sa1.sai aln_sa2.sai} {read1.fq read2.fq} > aln-pe.sam
```



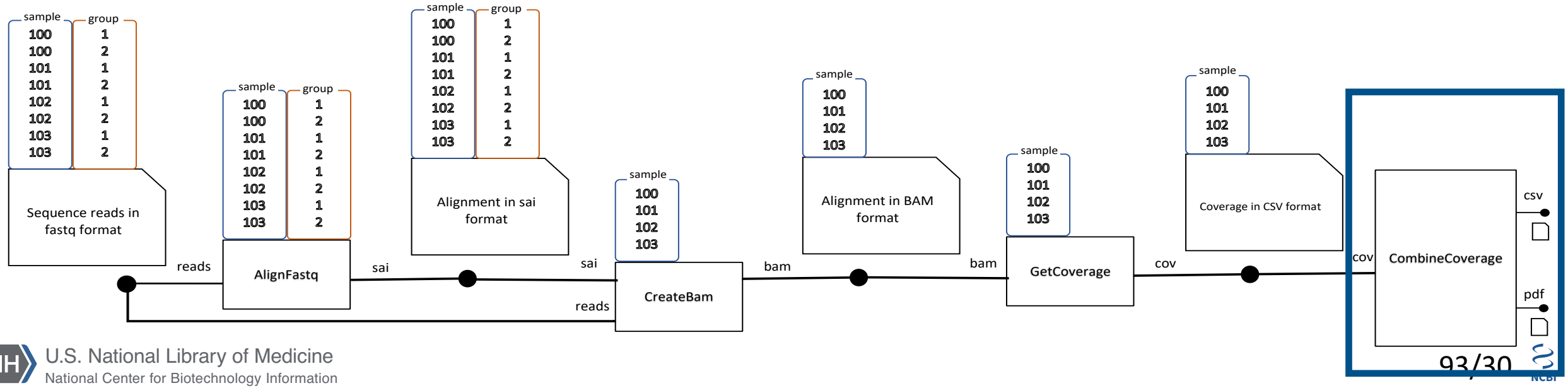
# GetCoverage Task in JUDI

```
class GetCoverage(Task):
  param = pdb_sample
  inputs = {'bam': f_bam}
  targets = {'cov': f_cov}
  actions = [('(echo coverage; samtools rmdup {} - | samtools mpileup - '
              '| cut -f4) > {}', ['$bam','$cov'])]
```



# CombineCoverage Task in JUDI

```
class CombineCoverage(Task):
    inputs = {'cov': f_cov}
    targets = {'csv': f_cov_combined, 'pdf': f_plt_combined}
    actions = [(combine_csvs, ['#cov', '#csv']),
                                                    ]
```

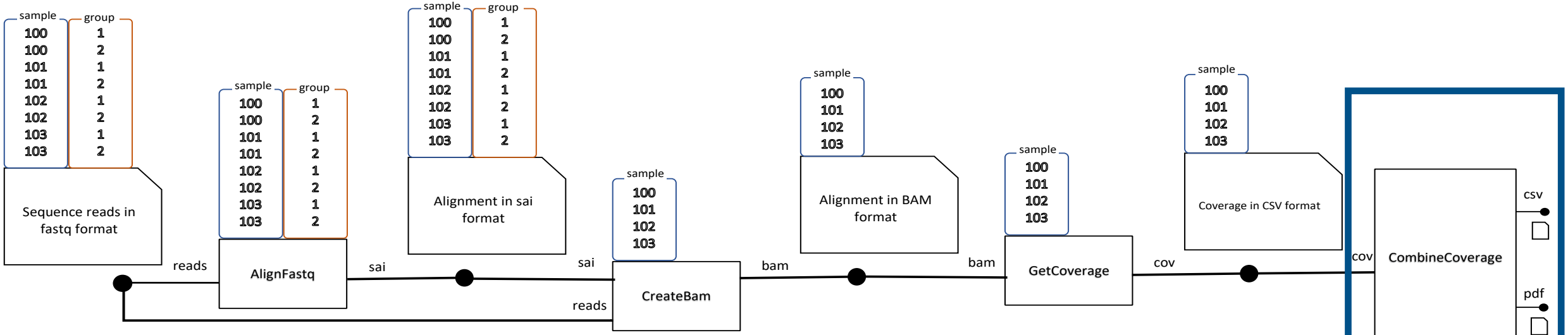


# CombineCoverage Task in JUDI

#file:
paramdb
for the file

$file:
path(s)

```
class CombineCoverage(Task):
    inputs = {'cov': f_cov}
    targets = {'csv': f_cov_combined, 'pdf': f_plt_combined}
    actions = [(combine_csvs, ['#cov', '#csv']),
                                                    ]
```

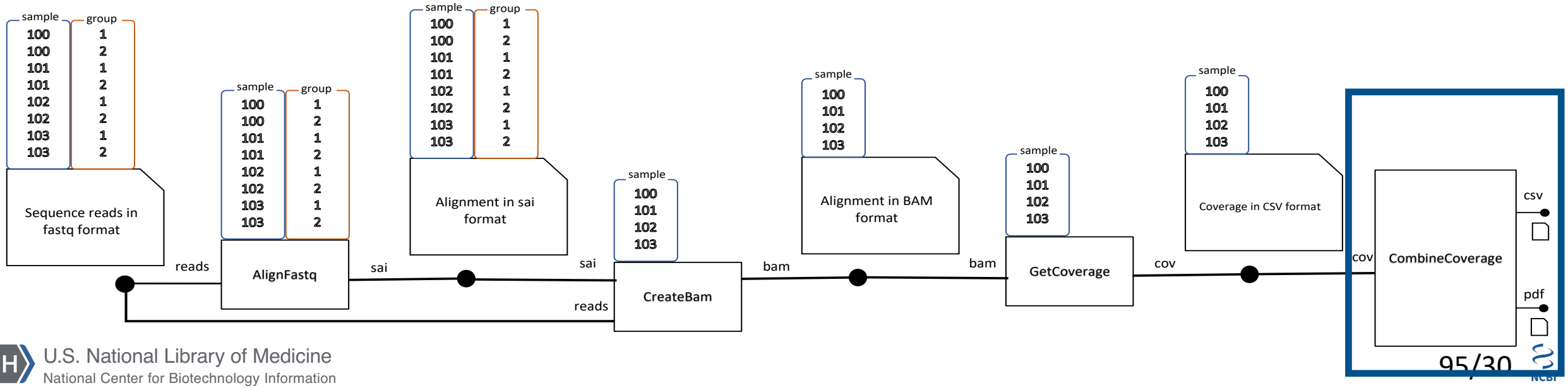


# CombineCoverage Task in JUDI

#file:
paramdb
for the file

$file:
path(s)

```
class CombineCoverage(Task):
    inputs = {'cov': f_cov}
    targets = {'csv': f_cov_combined, 'pdf': f_plt_combined}
    actions = [(combine_csvs, ['#cov', '#csv']),
               ("""echo "library(ggplot2); pdf('{}')
                ggplot(read.csv('{}'), aes(x = coverage)) + xlim(0,5) +
                geom_bar(aes(fill = factor(sample)), position ='dodge')"\
                | R --vanilla""", ['$pdf','$csv'])]
```



# Complete code & execution

dodo.py

```
from judi import File, Task, combine_csvs
from judi.paramdb import ParamDb

pdb_sample_grp = ParamDb('sample_grp')
pdb_sample_grp.add_param('100 101 102 103'.split(), 'sample')
pdb_sample_grp.add_param('1 2'.split(), 'group')


pdb_sample = ParamDb('sample')
pdb_sample.add_param('100 101 102 103'.split(), 'sample')
pdb_sample.show()


REF = 'hg_refs/hg19.fa'


path_gen = lambda x: '{}_{}.fq'.format(x['sample'],x['group'])
f_reads = File('seq', param = pdb_sample_grp, root = '.', path = path_gen)

f_aln = File('aln.sai', param = pdb_sample_grp)

f_bam = File('aln.bam', param = pdb_sample)
f_cov = File('cov.csv', param = pdb_sample)

f_cov_combined = File('combined.csv')
f_plt_combined = File('plot.pdf', root = '.')
```

```
class AlignFastq(Task):
  param = pdb_sample_grp
  inputs = {'reads': f_reads}
  targets = {'sai': f_aln}
  actions = [('bwa aln {} {} > {}', [REF,'$reads','$sai'])]

class CreateBam(Task):
  param = pdb_sample
  inputs = {'reads': f_reads, 'sai': f_aln}
  targets = {'bam': f_bam}
  actions = [('bwa sampe {} {} {} | samtools view -Sbh - '
              '| samtools sort - > {}', [REF,'$sai','$reads','$bam'])]

class GetCoverage(Task):
  param = pdb_sample
  inputs = {'bam': f_bam}
  targets = {'cov': f_cov}
  actions = [('(echo coverage; samtools rmdup {} - | samtools mpileup - '
              '| cut -f4) > {}', ['$bam','$cov'])]

class CombineCoverage(Task):
  inputs = {'cov': f_cov}
  targets = {'csv': f_cov_combined, 'pdf': f_plt_combined}
  actions = [(combine_csvs, ['#cov', '#csv']),
             ("""echo "library(ggplot2); pdf('{}')
              ggplot(read.csv('{}'), aes(x = coverage)) + xlim(0,5) +
              geom_bar(aes(fill = factor(sample)), position ='dodge')"\
              | R --vanilla""", ['$pdf','$csv'])]
```

Execute the pipeline

```
$ doit -f dodo.py
```

# List pipeline stages & check the status

# List pipeline stages & check the status

List of Tasks in a pipeline

```
$ doit list -f dodo.py
AlignFastq
CombineCoverage
CreateBam
GetCoverage
```

# List pipeline stages & check the status

List of Tasks in a pipeline

```
$ doit list -f dodo.py
AlignFastq
CombineCoverage
CreateBam
GetCoverage
```

Subtasks under different parameter settings

```
$ doit list --all --status -f dodo.py
R AlignFastq
U AlignFastq:group~1,sample~100
U AlignFastq:group~1,sample~101
U AlignFastq:group~1,sample~102
U AlignFastq:group~1,sample~103
U AlignFastq:group~2,sample~100
U AlignFastq:group~2,sample~101
U AlignFastq:group~2,sample~102
U AlignFastq:group~2,sample~103
R CombineCoverage
U CombineCoverage:
R CreateBam
U CreateBam:sample~100
U CreateBam:sample~101
U CreateBam:sample~102
U CreateBam:sample~103
R GetCoverage
U GetCoverage:sample~100
U GetCoverage:sample~101
U GetCoverage:sample~102
U GetCoverage:sample~103
```

# List pipeline stages & check the status

List of Tasks in a pipeline

```
$ doit list -f dodo.py
AlignFastq
CombineCoverage
CreateBam
GetCoverage
```

Subtasks under different parameter settings

```
$ doit list --all --status -f dodo.py
R AlignFastq
U AlignFastq:group~1,sample~100
U AlignFastq:group~1,sample~101
U AlignFastq:group~1,sample~102
U AlignFastq:group~1,sample~103
U AlignFastq:group~2,sample~100
U AlignFastq:group~2,sample~101
U AlignFastq:group~2,sample~102
U AlignFastq:group~2,sample~103
R CombineCoverage
U CombineCoverage:
R CreateBam
U CreateBam:sample~100
U CreateBam:sample~101
U CreateBam:sample~102
U CreateBam:sample~103
R GetCoverage
U GetCoverage:sample~100
U GetCoverage:sample~101
U GetCoverage:sample~102
U GetCoverage:sample~103
```

More details on **`doit`** manual
https://pydoit.org/contents.html

# Clean up files

# Clean up files

Cleanup all targets of a JUDI Task

```
$ doit clean -f dodo.py CombineCoverage
CombineCoverage: - removing file './judi_files/combined.csv'
CombineCoverage: - removing file './plot.pdf'
```

# Clean up files

Cleanup all targets of a JUDI Task

```
$ doit clean -f dodo.py CombineCoverage
CombineCoverage: - removing file './judi_files/combined.csv'
CombineCoverage: - removing file './plot.pdf'
```

Cleanup targets of a JUDI Task for a parameter setting

```
$ doit clean -f dodo.py GetCoverage:sample~101
GetCoverage:sample~101 - removing file './judi_files/sample~101/cov.csv'
```

# Parallel execution on multi-core

```
$ doit -n 8 -f dodo.py
```

More details and other commands on **`doit`** manual

# Parallel execution on HPC (biowulf)

```
class  AlignFastq(Task):
  inputs = {'reads': File('orig_fastq', path = path_gen)}
  targets = {'sai': File('aln.sai')}
  actions = [('srun -n 1 -N 1 -c 1'
              'bwa aln {} {} > {}', [REF, '$reads', '$sai'])]
```

biowulf$ doit -n 8 -f dodo.py

# Comparison with Snakemake

| Feature | Snakemake | JUDI |
|---|---|---|
| Parameter Database | Wildcard chars | Task and File level parameter setting using database concepts |
| File path handler | Manual | Taken care of by JUDI |
| Automatic collation of results | No such support | Supported |
| Multi-processing & HPC | Supported | Supported |
| Containerized storage and cloud execution | Supported | Future work |

# Availability

# Availability

- JUDI source code, Python PIP module, documentation

  https://github.com/ncbi/JUDI

# Availability

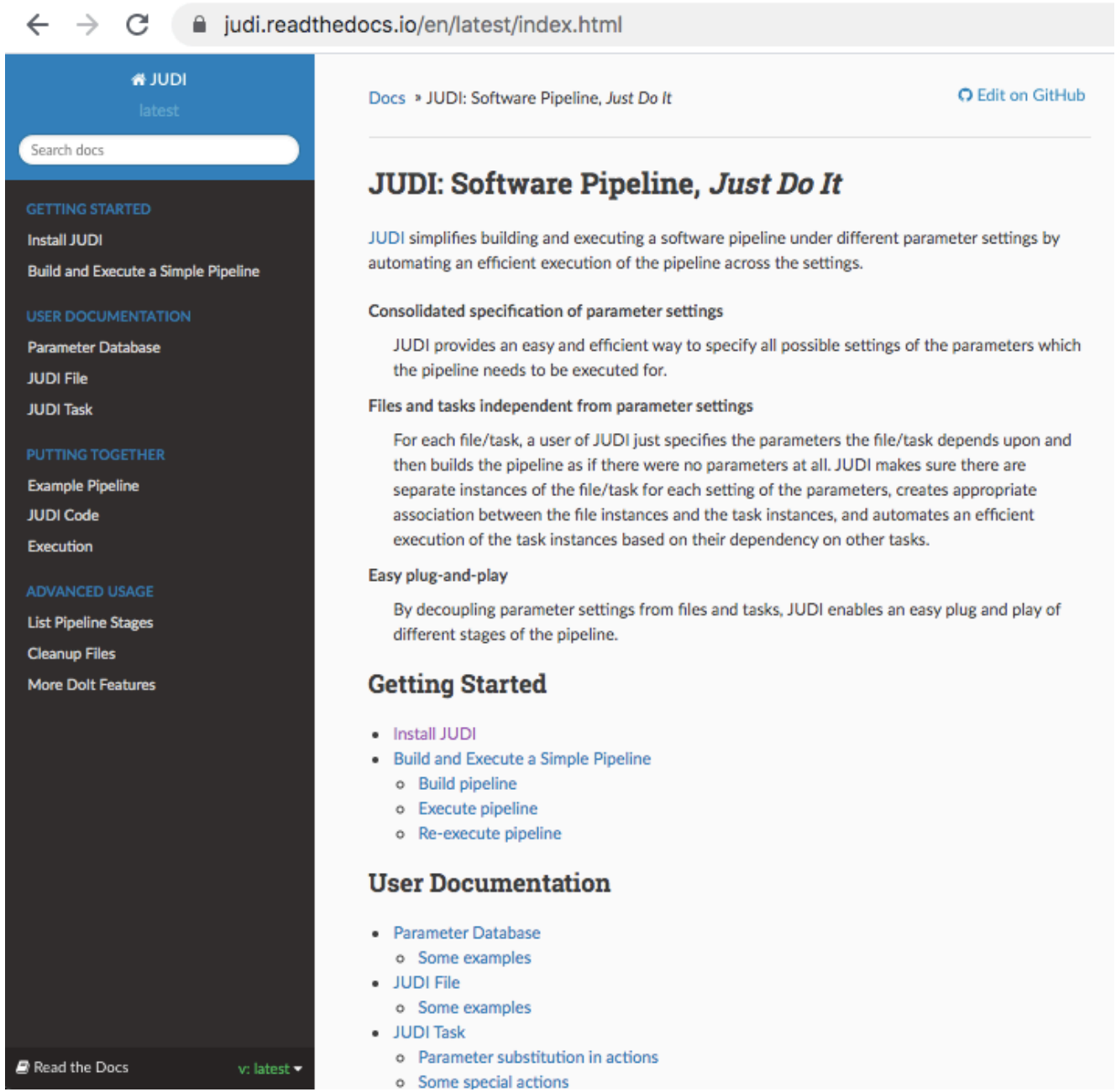
- JUDI source code, Python PIP module, documentation
  https://github.com/ncbi/JUDI

- Oxford Bioinformatics paper
  Soumitra Pal, Teresa M Przytycka, “Bioinformatics Pipeline using JUDI: *Just Do It”, Bioinformatics*, 2020, btz956

# Availability

- JUDI source code, Python PIP module, documentation
  https://github.com/ncbi/JUDI

- Oxford Bioinformatics paper
  Soumitra Pal, Teresa M Przytycka, “Bioinformatics Pipeline using JUDI: *Just Do It”, Bioinformatics*, 2020, btz956




https://judi.readthedocs.io

# Conclusions and future work

# Conclusions and future work

- Workflow manager that supports parameterization of files and tasks

# Conclusions and future work

- Workflow manager that supports parameterization of files and tasks
- Helps optimal execution and storage

# Conclusions and future work

- Workflow manager that supports parameterization of files and tasks
- Helps optimal execution and storage
- Provides library of collation functions

# Conclusions and future work

- Workflow manager that supports parameterization of files and tasks
- Helps optimal execution and storage
- Provides library of collation functions

- Need to catch up with other features (co-developers welcome!)

# Conclusions and future work

- Workflow manager that supports parameterization of files and tasks
- Helps optimal execution and storage
- Provides library of collation functions

- Need to catch up with other features (co-developers welcome!)
- Try with out-house projects (beta-testers welcome!)

# Conclusions and future work

- Workflow manager that supports parameterization of files and tasks
- Helps optimal execution and storage
- Provides library of collation functions

- Need to catch up with other features (co-developers welcome!)
- Try with out-house projects (beta-testers welcome!)
- SnakeJudi ?? (JUDI on top of Snakemake)

# Acknowledgements

National Center for Biotechnology Information
NCBI

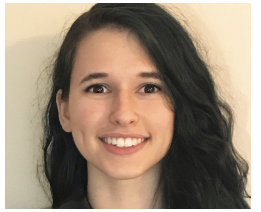
Teresa M. Przytycka

Damian Wojtowicz

Yoo-Ah Kim

Jan Hoinka

Bayarbaatar Amgalan



Ariella Saslafsky

Niko Darby