Stepwise imputation up to sequence level and resource allocation

Martina Franz

Overview

- Part 1: Plan imputation strategy
    - Direct or step-wise imputation
    - Handel duplicate samples
    - Impute up to sequence level
- Part 2: Resource allocation and slurm profiling
    - Factors impact on resource demand
    - Resource allocation
    - Slurm profiling
- Summary

Part 1: Plan imputation strategy

Scenario:

- Genotype data on 3 panels (partially overlapping):
    - Panel 1
    - Panel 2
    - Panel 3
- Aim: Impute panels 2 and 3 up to panel 1
    - Check panel overlap
    - Filter for unique samples

Plan imputation strategy – check sample overlap 1

Reference panel 1	X X X X – X X – – X – – – X – X X X
Study panel 2	X – X – X – – – – X X – X – X – X X
Study panel 3	X X – – X – X – – – X – X – – – x –

X : genotype available at that position
– : no genotype available at that position

Plan imputation strategy – check sample overlap 1

		Overlap study panel with reference panel:
Reference panel 1	X X X X – X X – – X – – – X – X X X	
Study panel 2	X – X – X – – – – X X – X – X – X X	5 position
Study panel 3	X X – – X – X – – – X – X – – – x –	4 positions

Plan imputation strategy – check sample overlap 1

		Number of samples:	Unique sample (no overlap with reference):
Reference panel 1	X X X X – X X – – X – – – X – X X X	95	
Study panel 2	X – X – X – – – – X X – X – X – X X	139	136
Study panel 3	X X – – X – X – – – X – X – – – x –	118	74

Questions to ask:

- Overlap study panels with reference panel

Plan imputation strategy – check sample overlap 1

		Number of samples:	**Unique sample (no overlap with reference):**
Reference panel 1	X X X X – X X – – X – – – X – X X X	95	
Study panel 2	X – X – X – – – – X X – X – X – X X	139	136
Study panel 3	X X – – X – X – – – X – X – – – x –	118	74

Questions to ask:

- Overlap study panels with reference panel
- Overlap study panels amongst one another
  - Case: multiple samples impute from different panels are wanted -> no further test needed

Plan imputation strategy – check sample overlap 1

		Number of samples:	Unique sample (no overlap with reference):
Reference panel 1	X X X X – X X – – X – – – X – X X X	95	
Study panel 2	X – X – X – – – – X X – X – X – X X	139	136
Study panel 3	X X – – X – X – – – X – X – – – x –	118	74

Questions to ask:

- Overlap study panels with reference panel
- Overlap study panels amongst one another
  - Case: only unique samples imputed to panel 1 -> create panel ranking list

Plan imputation strategy – check sample overlap 1

- Example create panel hierarchy:

1. Panel 1	keep all samples
2. Panel 3	keep samples not present on panel 1
3. Panel 2	keep samples not present on panel 1 and 3

- Subset samples according to panel hierarchy

Workflow unique samples only

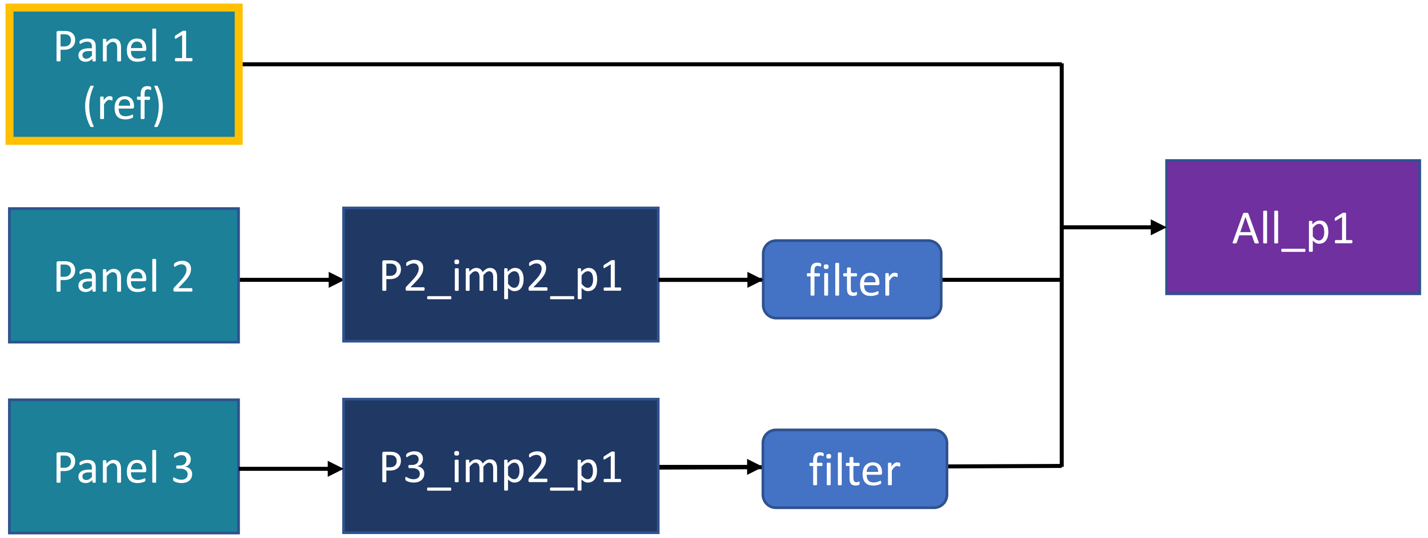
1. Create panel hierarchy
2. Subset samples according to panel hierarchy
3. Create references from panel 1
4. Impute panel 3 to reference panel 1 => panel3_impTo_panel1
5. Impute panel 2 to reference panel 1 => panel2_impTo_panel1
6. Merge panel_1, panel3_impTo_panel1, panel2_impTo_panel1
   => **all_panel1**

Workflow unique samples only

1. Create panel hierarchy
2. Subset samples according to panel hierarchy
3. Create references from panel 1
4. Impute panel 3 to reference panel 1 => panel3_impTo_panel1
    1. Mask poorly imputed variants, e.g. set variants to missing for DR2<0.9
5. Impute panel 2 to reference panel 1 => panel2_impTo_panel1
    1. Mask poorly imputed variants, e.g. set variants to missing for DR2<0.9
6. Merge panel_1, panel3_impTo_panel1, panel2_impTo_panel1

   => **all_panel1**

Workflow unique samples + high quality imputed



Plan imputation strategy – check sample overlap 2

		Overlap study panel with reference panel:
Reference panel 1	X X X X – X X – – X – – – X – X X X	
Study panel 2	– – – – X – – X X X X – X – X – – –	1 position
Study panel 3	X – X – X – – – – X X – X – X – X X	5 positions

Step-wise imputation:

Study panel 2

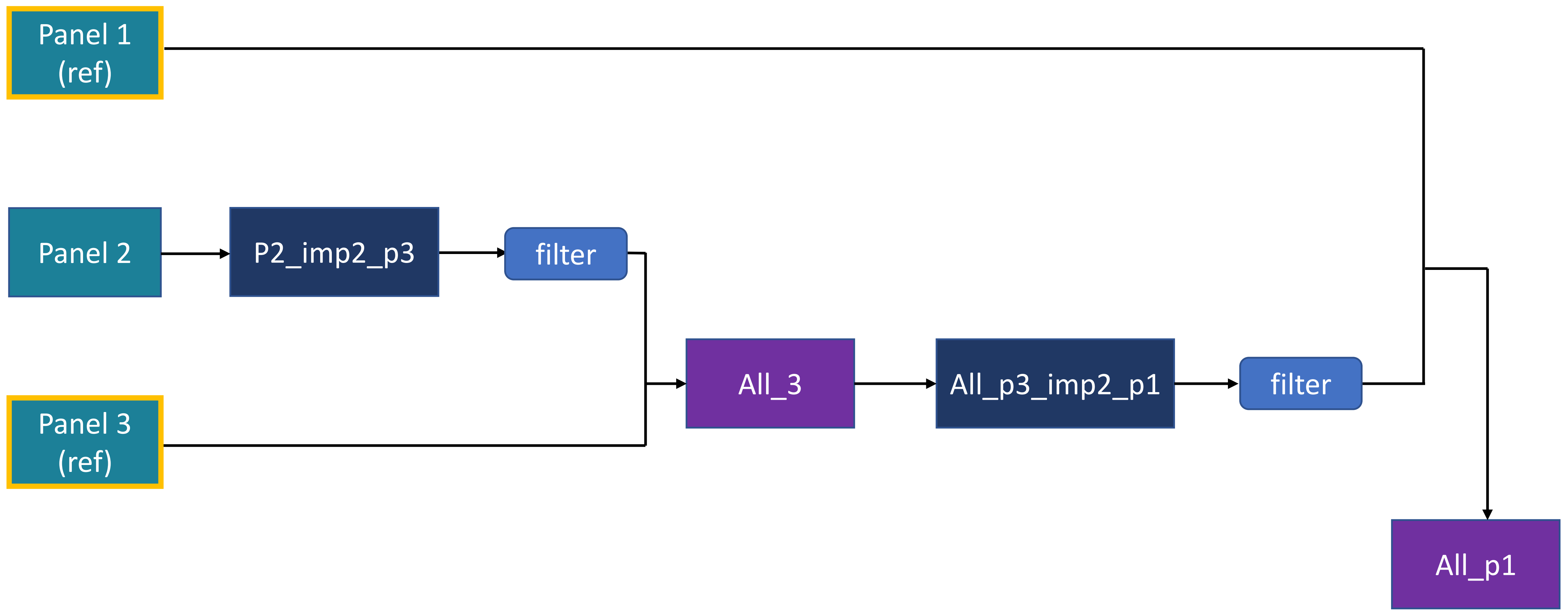
Test: - create reference from panel 3
- impute panel 2 -> panel 3 -> panel 1

Workflow unique samples + high quality imputed

1. Create panel hierarchy
2. Subset samples according to panel hierarchy
3. Create references from panel 1 and panel 3
4. Impute panel 2 to reference panel 3 => panel2_impTo_panel3
    1. Mask poorly imputed variants, e.g. set variants to missing for DR2<0.9
5. Merge panel2_impTo_panel3 with panel 3 => all_panel3
6. Impute all_panel3 to panel 1 => all_panel3_impTo_panel1
    1. Mask poorly imputed variants, e.g. set variants to missing for DR2<0.9
7. Merge panel_1, all_panel3_impTo_panel1

    => **all_panel1**

Workflow step-wise imputation



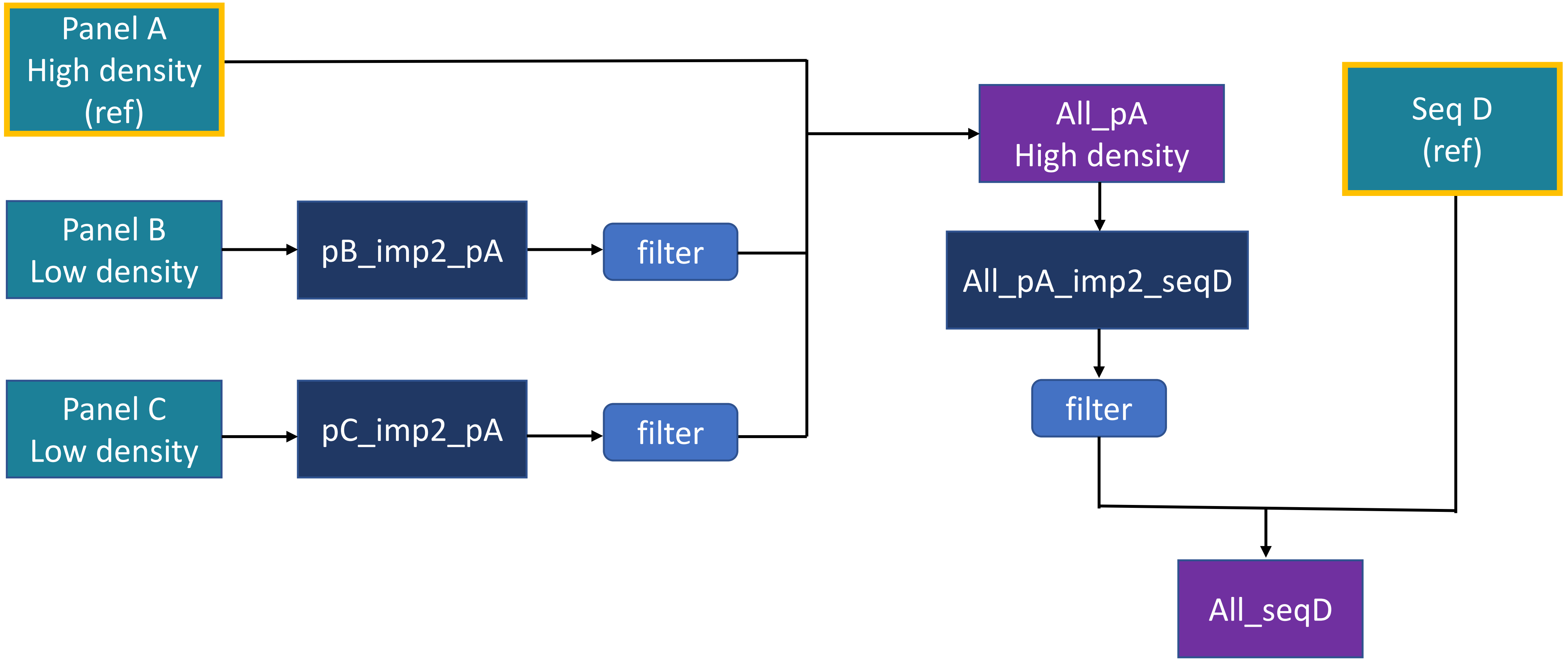
Part 1: Plan imputation strategy up to seq level

Scenario:

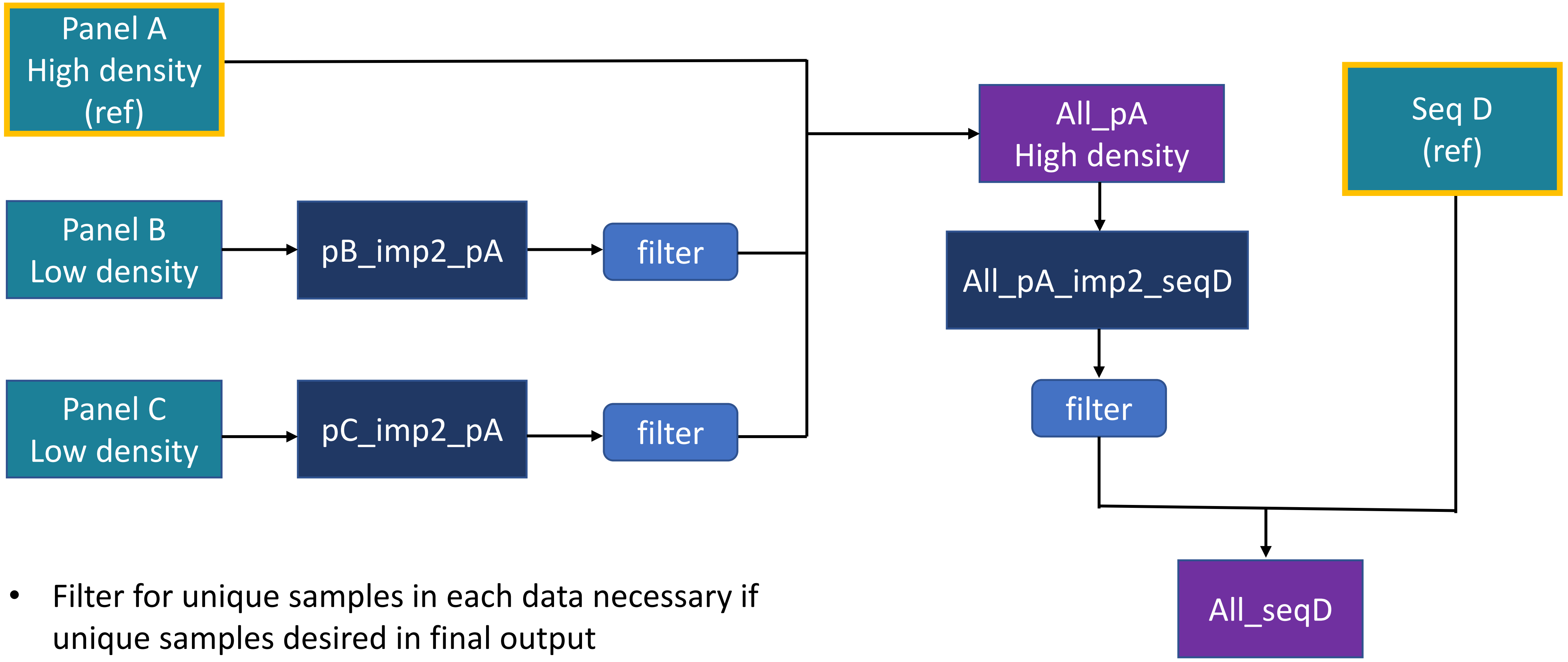
- Sequence data SeqD
- Genotype data on 3 panels:
    - Panel A (high density)
    - Panel B (low density)
    - Panel C (low density)

- Aim: Impute panels A-C up to sequence level
    - Check panel overlap
    - Filter for unique samples

Stepwise imputation up to sequence level



Stepwise imputation up to sequence level



- Filter for unique samples in each data necessary if unique samples desired in final output
- Seq data are highest ranking dataset

Questions?

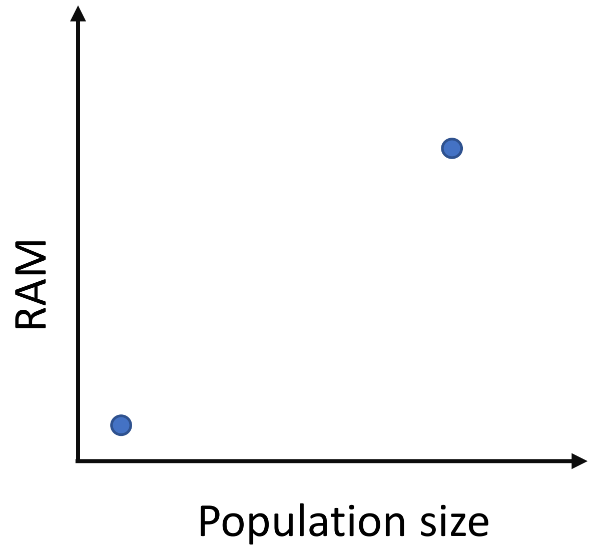
Part 2: Resource use

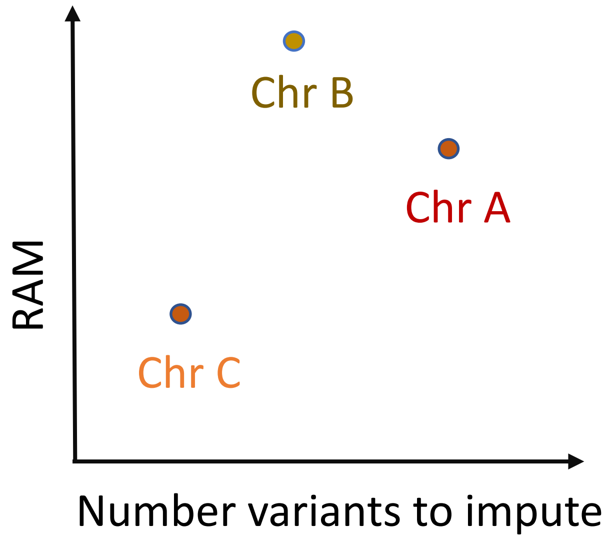
- Factors impacting on resource demand
- Resources allocation
- Slurm profiling

Main factors impacting on resource demand

- Population size of samples to impute
- Number of variants to impute
- Number of chromosomes to impute
- Window size
- Software used for imputation
  - Beagle: filetype of reference
    - Convert reference to bref3 format reduces memory requirement

Resource demand: population size and variants





- More samples take more RAM and compute time
  - If relationship is linear needs to get tested
- Resource demand does not necessarily increase linear with number of variants that get imputed
  - ‘tricky regions’ take up more compute time and RAM than ‘easy regions’

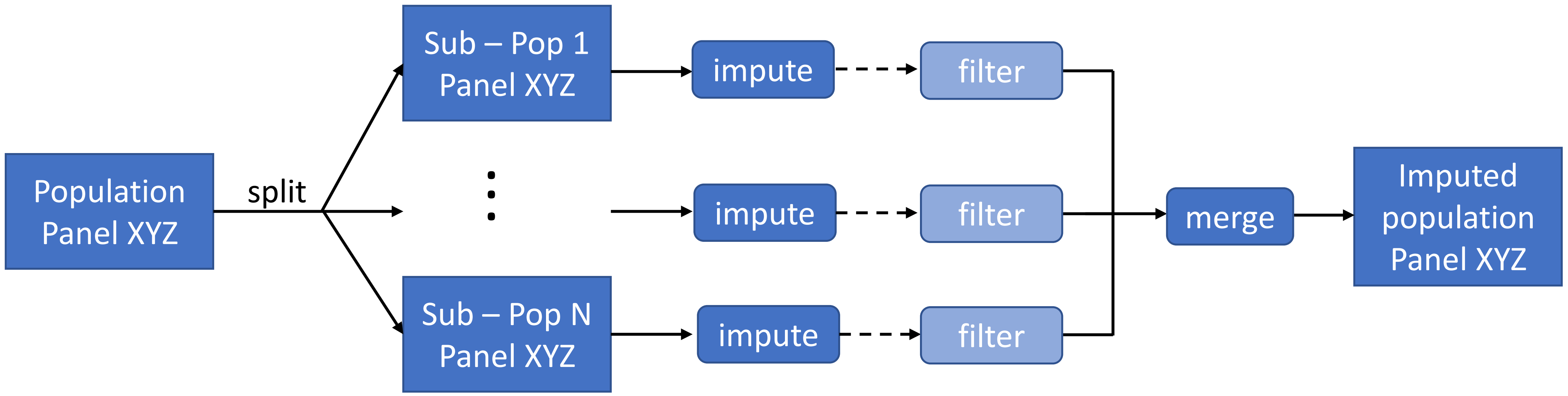
Manage high resource demand

- Split population into chunks
  - Smaller demand of RAM per job 👍
  - Shorter runtime per job 👍
  - More jobs to run 👎
  - Decrease required amount of RAM might enable use of less crowded partitions 👍
  - Overall resource demand needs to get tested

Info on NESI slurm partitions:

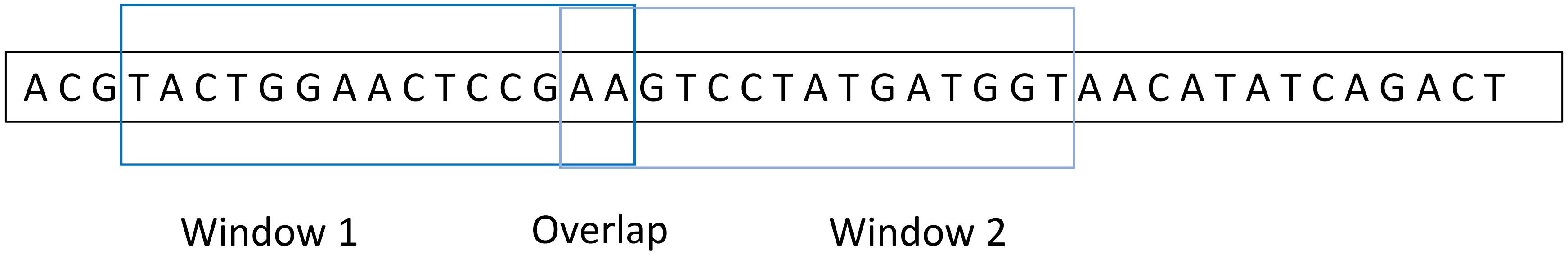
https://support.nesi.org.nz/hc/en-gb/articles/360000204076-Mahuika-Slurm-Partitions

Split population



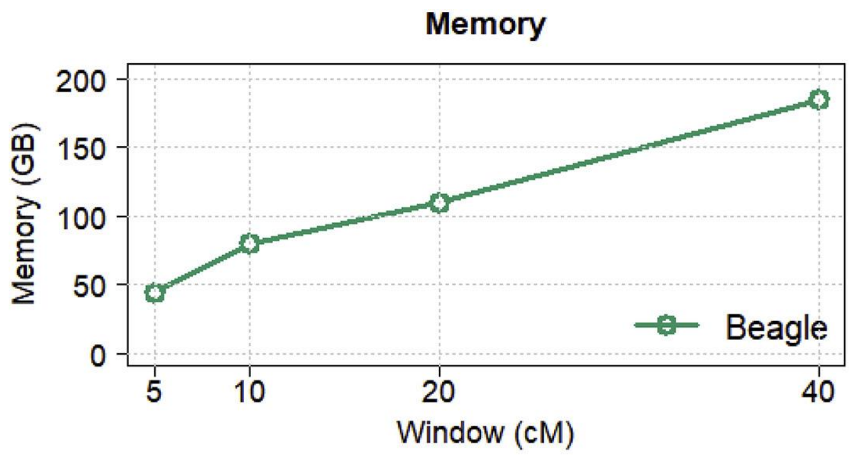
- **Important**: split samples randomly to prevent stratification by age, breed, region, …
- Number of chunks to split depends on desired resource investment per job

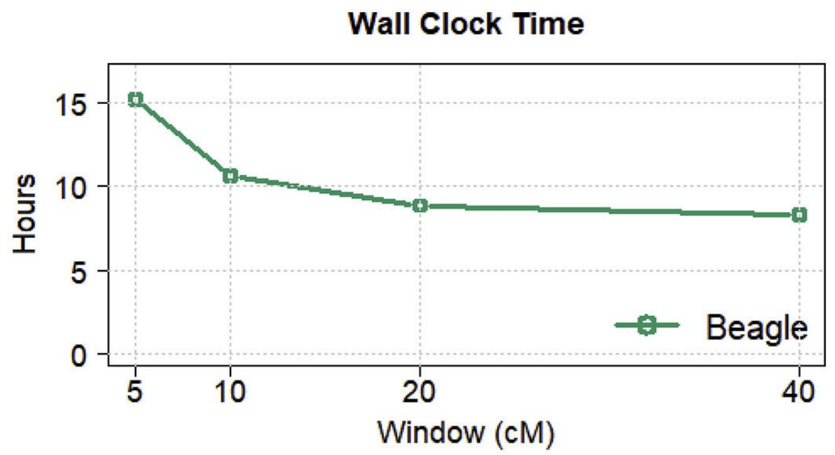
Window size


A C G T A C T G G A A C T C C G A A G T C C T A T G A T G G T A A C A T A T C A G A C T
Window 1
Overlap
Window 2

- Window: specifies the cM length of each sliding window
  - beagle default value 40 cM (min. 1.1 times overlap)
- Overlap: specifies the cM overlap between sliding windows
  - beagle default value 2.0 cM

Impact of window size on phasing





Browning et al. 2021, Fig. 4

Decrease of window size can reduce memory demand

Run jobs and specify resources

- Submit slurm job(s) to run sub-setting, imputation, filtering, merging, …
  - Write slurm script, or
  - Submit bash script to slurm work load manager
- Specify resources for slurm job
  - --mem: memory (RAM) per node
  - --time: maximum total run time
  - --cpus-per-task: number of processors per task (default 1)

Find good resource settings for slurm job

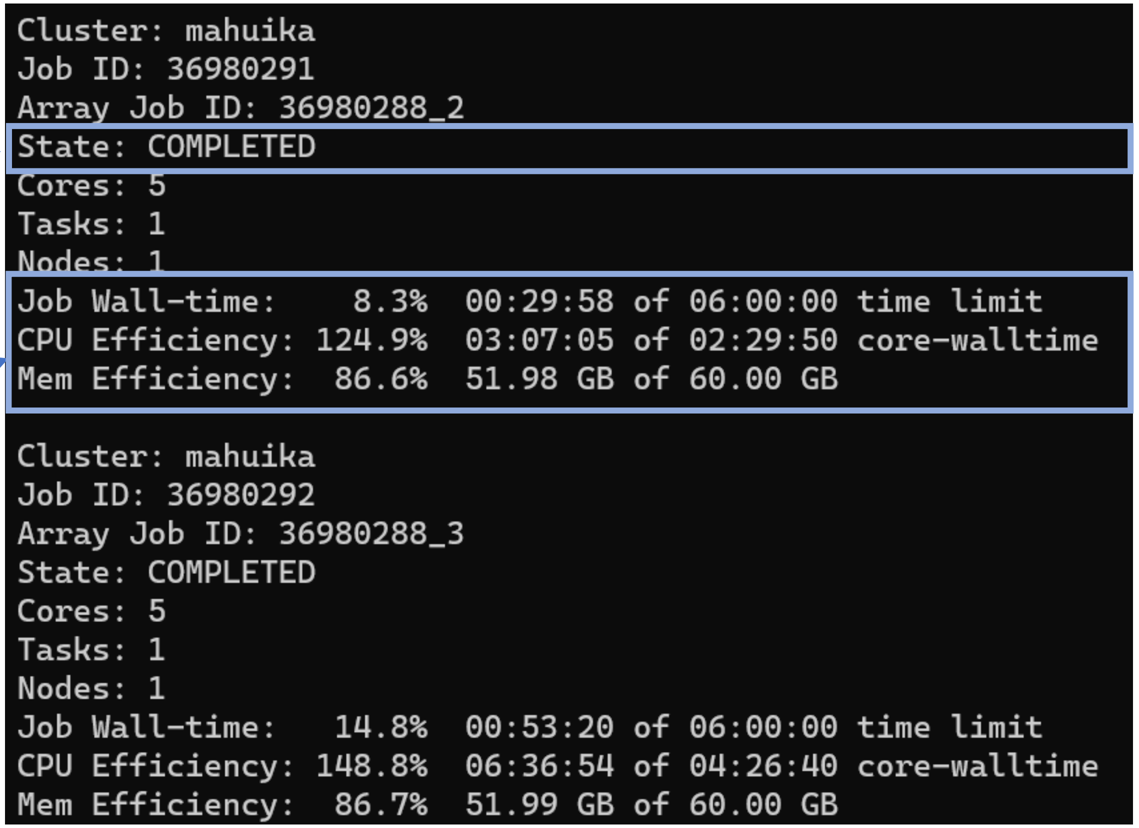
- Estimate potential resource demand based on prior experience
  - Allocate plenty of time so the job won’t time out
  - Estimate amount of RAM the job will need
  - If the submitted job runs a program that allows for parallelisation (e.g. bcftools and beagle) then specify number of CPUs
- Check on consumed resources of job:
  - ‘nn_seff JOBID’
- Adapt resource settings if job needs to run again + gain experience on what kind of job takes how much resources

Find good resource settings for slurm job

- Check on consumed resources by job:
  - ‘nn_seff JOBID’

Job status:
COMPLETED, FAILED,
OUT_OF_MEMORY ,
TIMEOUT

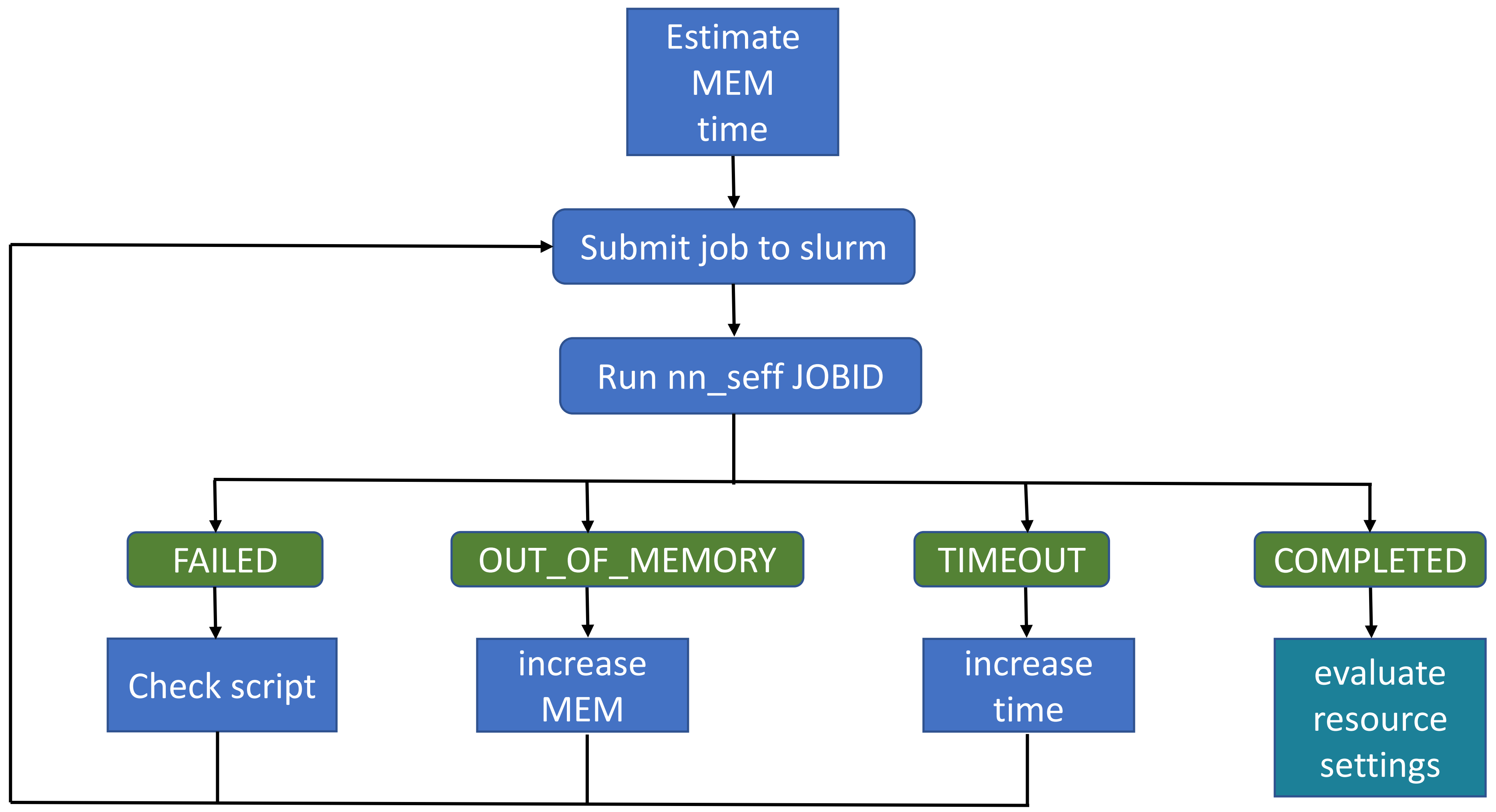
Job resource performance



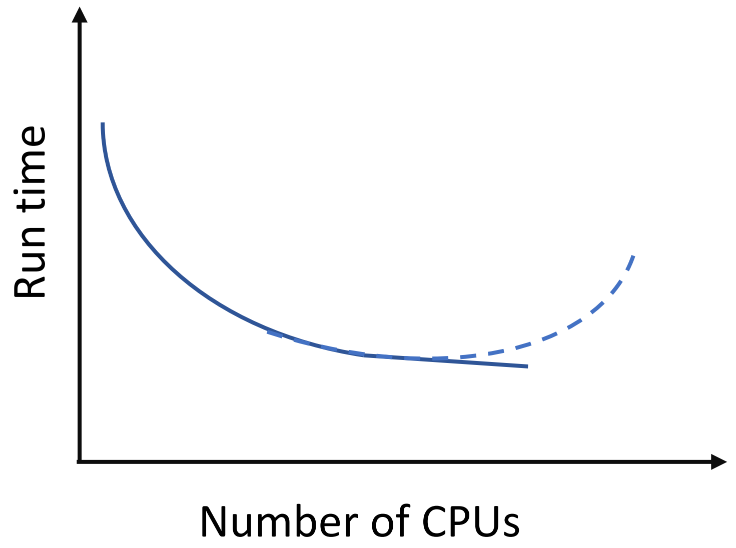
```
Cluster: mahuika
Job ID: 36980291
Array Job ID: 36980288_2
State: COMPLETED
Cores: 5
Tasks: 1
Nodes: 1
Job Wall-time:    8.3%  00:29:58 of 06:00:00 time limit
CPU Efficiency: 124.9%  03:07:05 of 02:29:50 core-walltime
Mem Efficiency:  86.6%  51.98 GB of 60.00 GB

Cluster: mahuika
Job ID: 36980292
Array Job ID: 36980288_3
State: COMPLETED
Cores: 5
Tasks: 1
Nodes: 1
Job Wall-time:   14.8%  00:53:20 of 06:00:00 time limit
CPU Efficiency: 148.8%  06:36:54 of 04:26:40 core-walltime
Mem Efficiency:  86.7%  51.99 GB of 60.00 GB
```

Workflow finding resource settings



Specifying the number of CPUs



- If overhead for parallelisation gets to big run time might increase again when specifying more CPUs
- **Run tests** to find sweet spot where one gets a strong increased in run time before saturation effect sets in

Slurm profiling

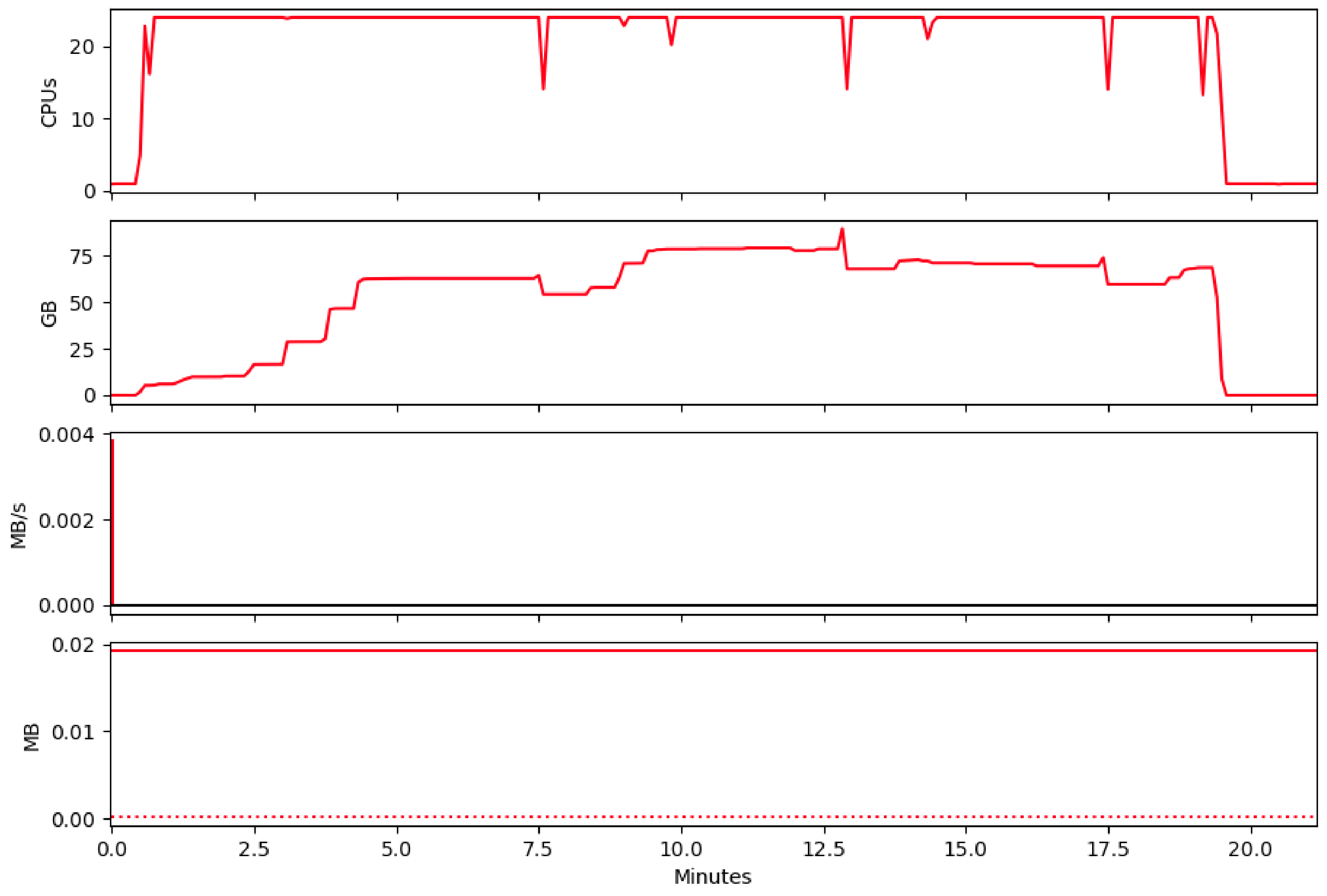
- Info on slurm profiling and access to plot script:
  https://genomicsaotearoa.github.io/Workshop-Bash_Scripting_And_HPC_Job_Scheduler/7_supplementary_2/
  Or
  https://support.nesi.org.nz/hc/en-gb/articles/360000810616-Slurm-Native-Profiling

- To run profiling add:
  ‘#SBATCH --profile task’ to slurm script, or
  ‘--profile task’ to sbatch command when submitting a bash script

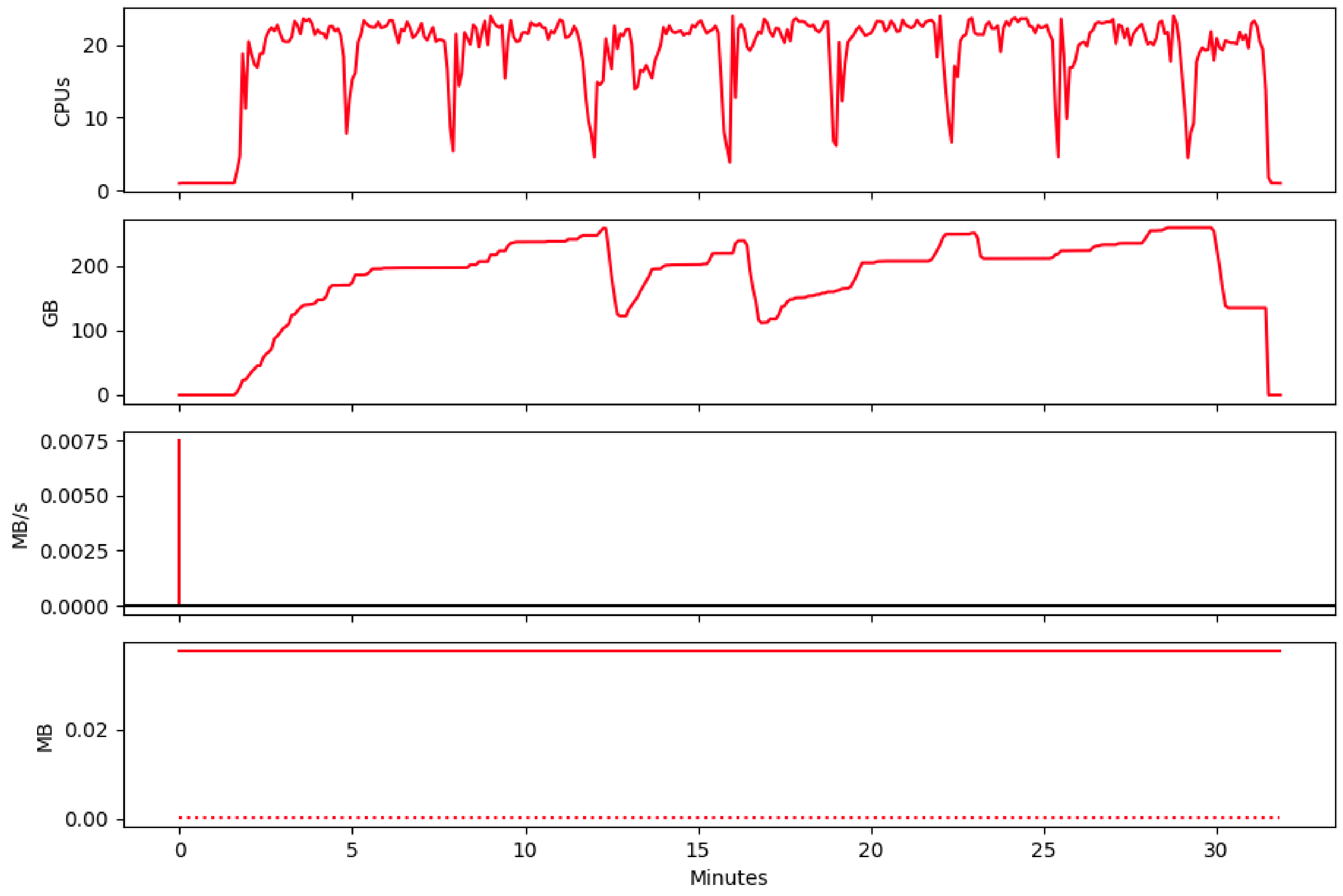
- Summarize output:
  ‘sh5util -j JOBID’

- Generate profile plot:
  ‘module purge; module load Python’
  ‘python profile_plot_Jul2020.py job_JOBID.h5 ‘

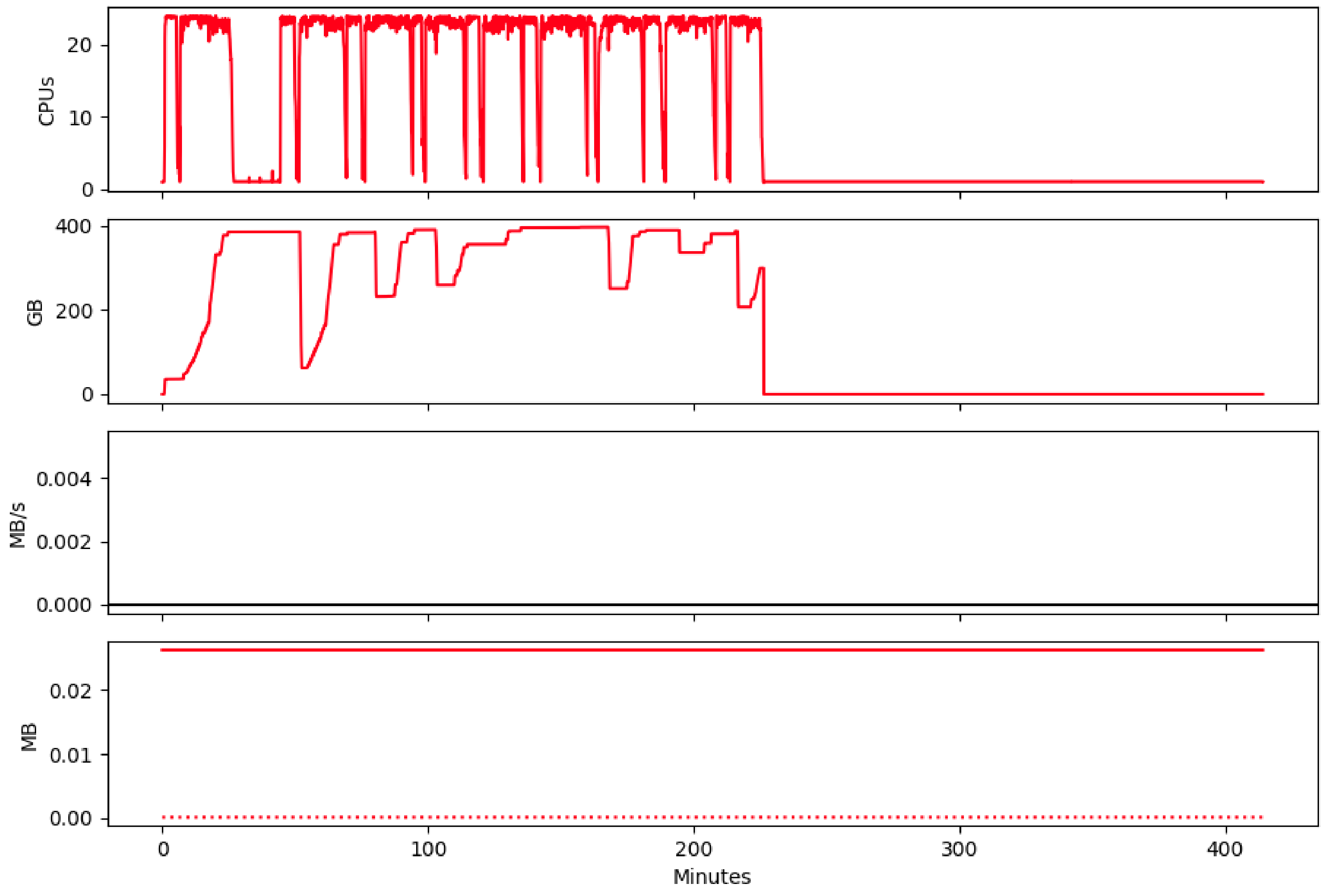
Profiling example output 1



Profiling example output 2



Profiling example output 3



Summary resource allocation

- Decrease resource demand:
  - Lower window size
  - Split population into chunks (increases number of jobs to run)
- Resource allocation:
  - Single/seldom run job: manual testing
  - Repeatedly run job: slurm profiling
- Run testing on longest chromosome (most variants to impute) and/or most complicated to impute chromosome

Questions?