GATK-Parabricks on Gadi @ National Computational Infrastructure

Summary

Prepared by

Locedie Mansueto, Ramil Mauleon
Southern Cross University, Lismore NSW Australia

Purpose of benchmarking

Optimising for computational performance of germline workflows for Parabricks (NVIDIA) and GATK (Broad Institute). The benchmarking was performed using Gadi High Performance Computing at the Australian National Computational Infrastructure (NCI).

Conclusion

Parabricks enabled fast alignment and variant discovery for large cohorts with low service units consumption. However the joint genotyping step is difficult to perform without the tool to merge GVCFs, and we need to use vanilla GATK to do the merging task. The Parabricks implementation of genotypegvcf is comparable in result with the vanilla GATK GenotypeGVCF, but is planned to be deprecated. The alternative glnexus is convenient to use and faster but has a lower discovery rate than vanilla GATK.

We recommend keeping the parabricks genotypegvcf and to implement a convenient tool to merge GVCFs as an alternative to GenomicsDB in GATK. A suggestion is for glnexus to also output the merged gvcf in addition to the final bcf, because non-variant information from the gvcfs cannot be recovered from the final bcf nor vcf.

Based on the work outlined below, at the optimal settings identified, the Parabricks germline pipeline is estimated to finish 15-20x faster, and uses 5x less service units than vanilla GATK.

The speed-up of SNP discovery using the GPU-assisted methods will greatly assist molecular geneticists, breeders, and diagnosticians who identify variants from large sets of samples under time constraints, wherein decisions on the next step of research or medical diagnostic for treatment rely on the timely availability of information of variants in samples.For example in breeding, SNP genotyping a population at F1 allows identification of true offsprings from a cross, since planting the next generation population is time-bound (the planting season cannot be delayed) and resource intensive. Another example in medical diagnosis: a quick turnaround time for variant identification is important for COVID-19 management.

Tools & workflows

List

Table 1. List and information about the variant calling pipelines used in the benchmarking exercise.

Title	Version	Maturity	Creators	Source	License	Workflow manager	Container
PB germline	3.1.6*	stable	NVIDIA	Parabricks	NVIDIA license*	None	Singularity
GATK germline	4.1.4.0	stable	Broad Institute	GATK4			

*Parabricks v3 is now deprecated and no longer supported or available for download. Parabricks v4 is now available, free to use for researchers, and with significant performance improvements.

Link to original workflow documentation

- GATK / PB germline pipeline at Gadi@NCI
- Parabricks Germline Pipeline
- GATK Germline Pipeline

Visual comparison (workflow diagrams)

Before running both GATK and Parabricks workflows, the dict and fai files are generated from the reference fasta file using: gatk CreateSequenceDictionary and samtools faidx (GATK prepare reference FASTA).

Vanilla GATK fastq2bam

The GATK fastq2bam consists of several steps to produce the alignment bam from the fastq and reference files.

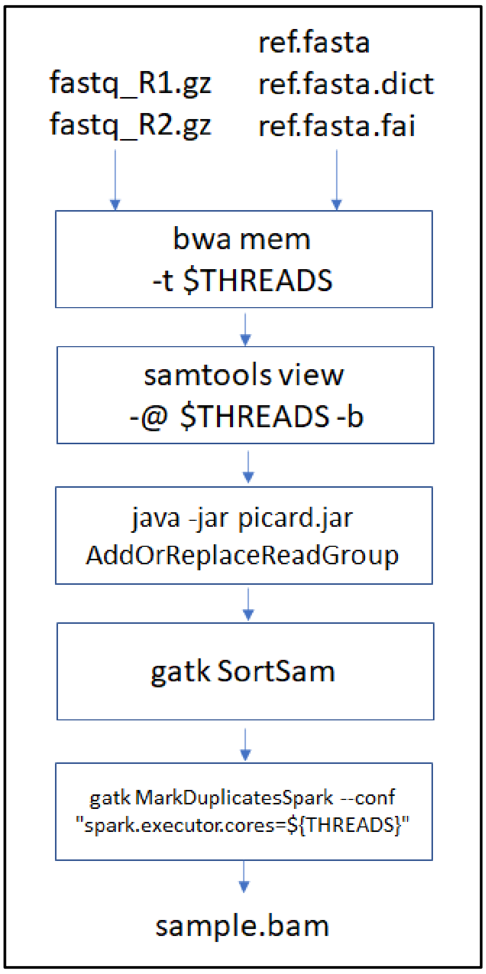

fastq_R1.gz
fastq_R2.gz
ref.fasta
ref.fasta.dict
ref.fasta.fai
bwa mem -t $THREADS
samtools view -@ $THREADS -b
java -jar picard.jar AddOrReplaceReadGroup
gatk SortSam
gatk MarkDuplicatesSpark --conf "spark.executor.cores=${THREADS}"
sample.bam

Figure 1. Diagram of the vanilla GATK fastq2bam workflow.

From parabricks documentation of fq2bam, the BQSR step is optional and performed only if --knownSites file is provided. Since we have no known sites for our species, BQSR is not implemented in the GATK vanilla workflow.

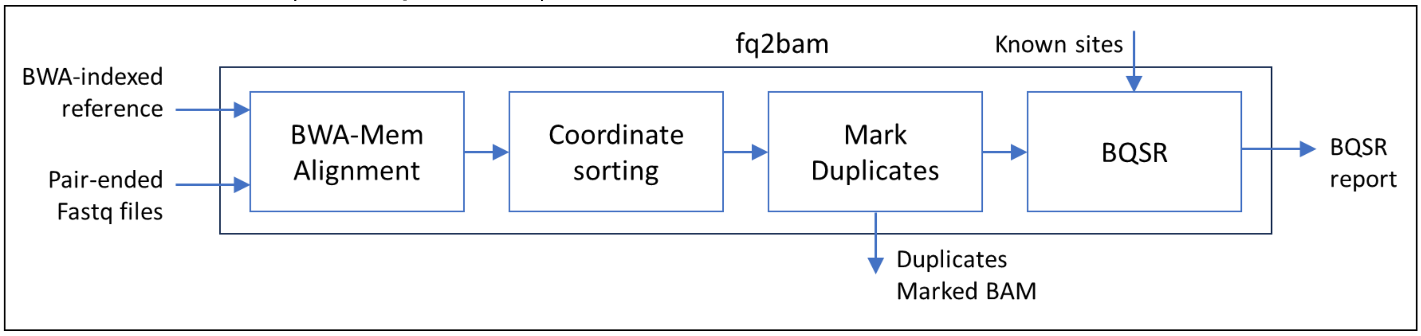


Figure 2. The fq2bam workflow, the BQSR step is skipped due to lack of gold standard SNPs in the species of interest. (adapted from https://docs.nvidia.com/clara/parabricks/v3.0/text/fastq_and_bam_processing.html#fq2bam)

Vanilla GATK HaplotypeCaller is called per chromosome using the `-L $CHR` option, where `$CHR` is the chromosome/contig name in the fasta file. All unassembled contigs are called together by excluding all assembled chromosomes with the option `-XL $CHR01 -XL $CHR02 .. -XL $CHRNN`

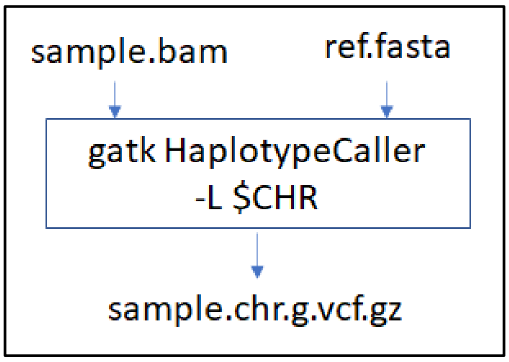


Figure 3. Diagram of the vanilla GATK HaplotypeCaller workflow.

Unlike in GATK HaplotypeCaller, which is run with 1 core per chromosome, Parabricks haplotypecaller is run for the entire genome using multiple cores. BQSR report is optional, and is not available for our species.

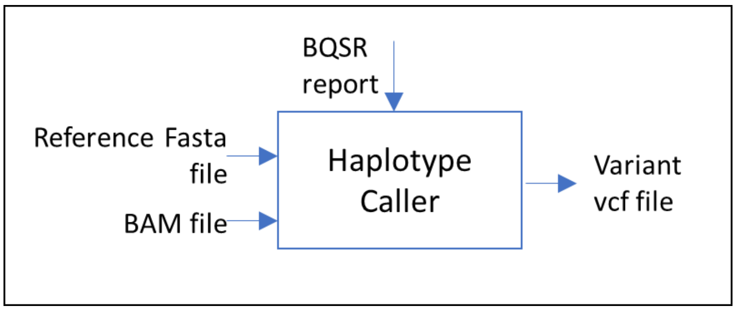


Figure 4. Diagram of the Parabricks haplotypecaller workflow (adapted from https://docs.nvidia.com/clara/parabricks/v3.0/text/variant_callers.html#haplotypecaller)

Merge gvcf for joint genotyping

GenomicsImportDB runs for more than 48 hours if all chromosomes are included. To avoid this, both vanilla GATK and Parabricks-generated g.vcf.gz files are imported to the database per chromosome: one set of database folders for GATK and another for the Parabricks results. Each chromosome creates a database folder with all the samples. SelectVariants generates a g.vcf.gz for each chromosome containing all samples. The GenotypeGVCFs performs the joint genotyping per chromosome.

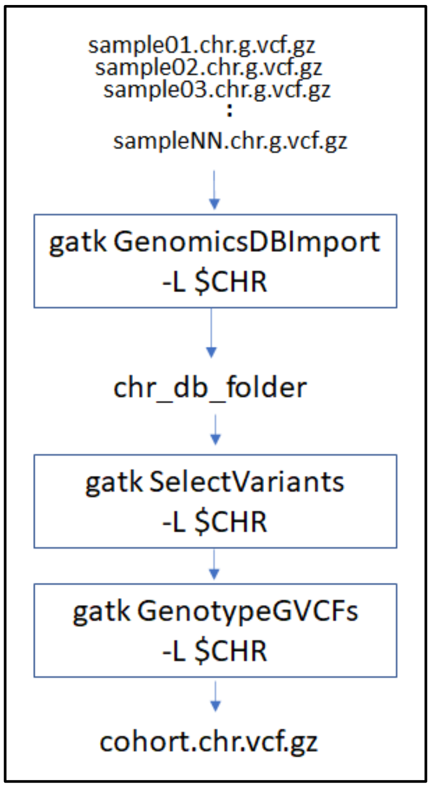


Figure 5. Diagram for vanilla joint genotyping using the GATK GenomicsDB database.

The final step is to combine all per-chromosome vcfs into one vcf containing all samples for the entire genome.

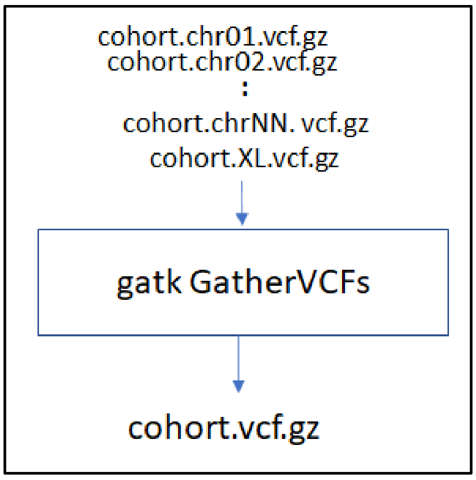


Figure 6. Concatenate all chromosomes into the final VCF file

Benchmarking: PB germline vs. GATK germline

Benchmark datasets

The datasets used for this benchmarking are 161 publicly available Cannabis paired-end whole genome sequence reads, including 55 from PRJNA310948 (Lynch et al., 2016), 40 from PRJNA575581 (McKernan et al., 2020), 2 from (Welling et al., 2020) (SRR12845227, SRR12845228), 58 from https://www.medicinalgenomics.com/kannapedia-fastq/, from Cannabis genome sequencing projects PK_SIL (SRR352161.2- SRR352169.2), CBDRx (ERR3850862), Finola (SRR7285294), JL (SRR10019825), and JD-DASH (RSP11074), Cannatonic (RSP11349) from https://www.medicinalgenomics.com/crypto-funded-public-genomics.

The reference genome used is Cannabis CBDRx (GCF_900626175.2).

Service unit optimization for vanilla GATK

First, optimization was performed for both GATK and Parabricks to determine the optimal number of cores and RAM to use to minimize service unit consumption. Samples of the datasets were used to cover a wide range of input file sizes, run with different numbers of cores and RAM.

For the fastq2bam pipeline (GATK), bwa mem and MarkDuplicate have an option for multi-threading. Core numbers were varied to obtain the setting which consumes the fewest minimum service units. **Based on the figure below, 10 cores were selected as optimum.**

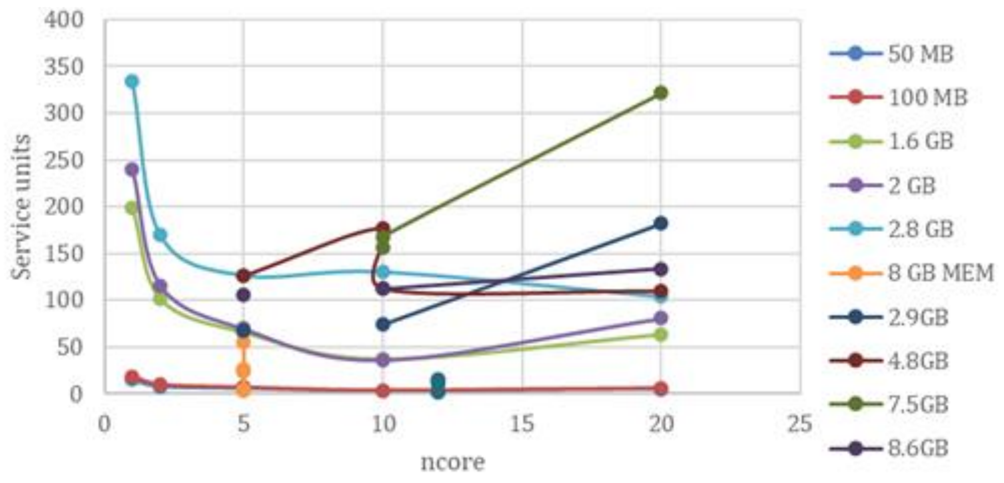


Figure 7. Plot of service units used by the GATK fastq2bam workflow (Figure 1) when assigned different numbers of cores and RAM usage.

GATK HaplotypeCallller does not have an option for multi-threading, but it is possible to split the genome into intervals and then distribute each interval across different cores. When used with 1 core only, fastq inputs above 1 GB will take more than 48 hours to run, exceeding the normal maximum walltime for Gadi jobs. We chose to split variant-calling to a thread per chromosome, plus 1 thread for all the unassembled contigs. For the task of assigning 1 job per chromosome, different memory sizes were compared for different input data sizes. As shown in the figure below, service units do not vary much for memory sizes in the range 1 to 16 GB.

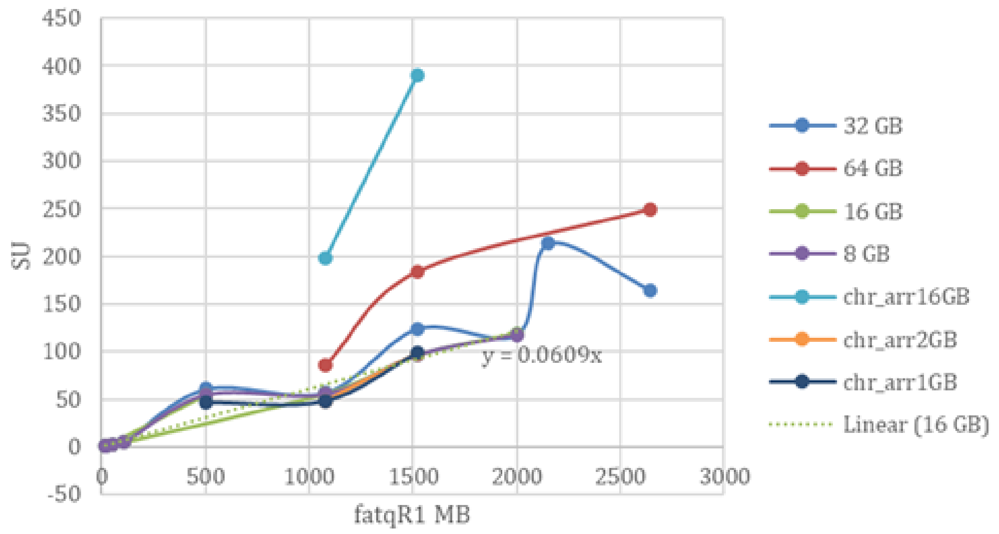


Figure 8. Plot of service units used for GATK HaplotypeCaller workflow (Figure 3). Values are the totals for 13 cores, 1 core per chromosome and 1 for all unassembled contigs for different Fastq input sizes.

Service units optimization for Parabricks

To optimise the Parabricks runs, different numbers of GPU and CPU cores were tried. Each GPU needs 12 CPU cores: therefore runs with different input file sizes were compared using 1, 2 or 3 GPUs, with the corresponding 12, 24 or 36 CPUs. The results are illustrated in the figure below and these are the observations:

For fq2bam, as NGPU increases (and NCPU) from 1 to 2, the wall time improves and SU consumption remains the same.

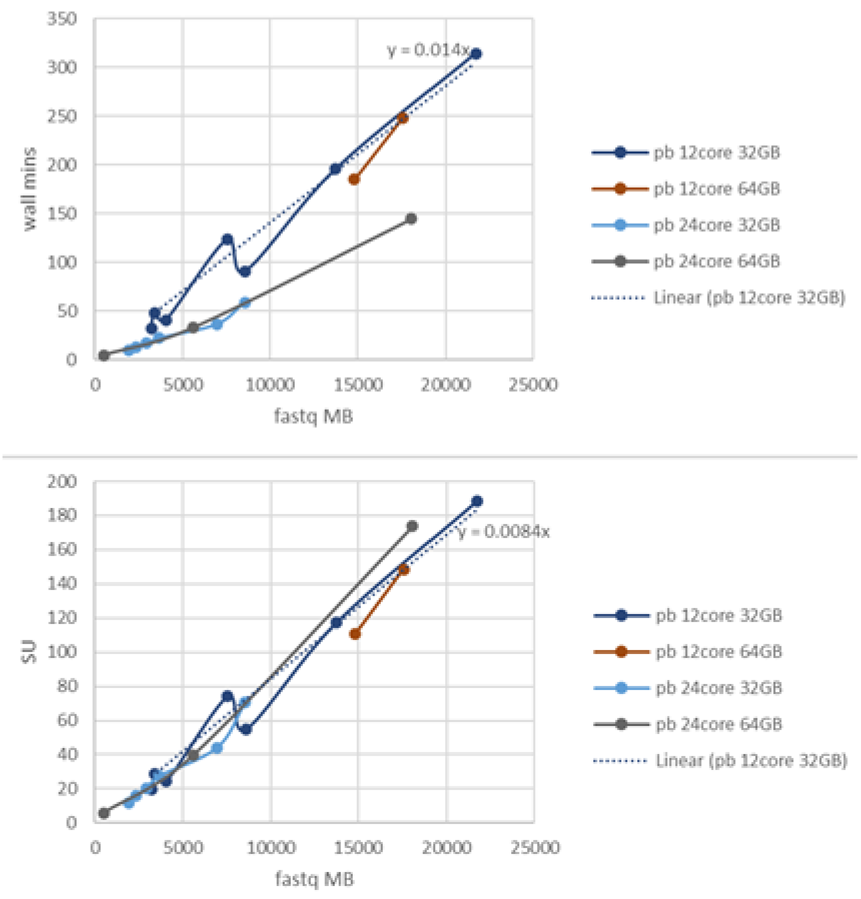


Figure 9. Parabricks fq2bam workflow (Figure 2) walltime and service units consumption for different numbers of cores and assigned RAM.

For haplotypecaller, as NGPU increases (and NCPU) from 1 to 2, wall time improves and SU decreases. SU remains the same with a slight wall time improvement when increasing NGPU further.

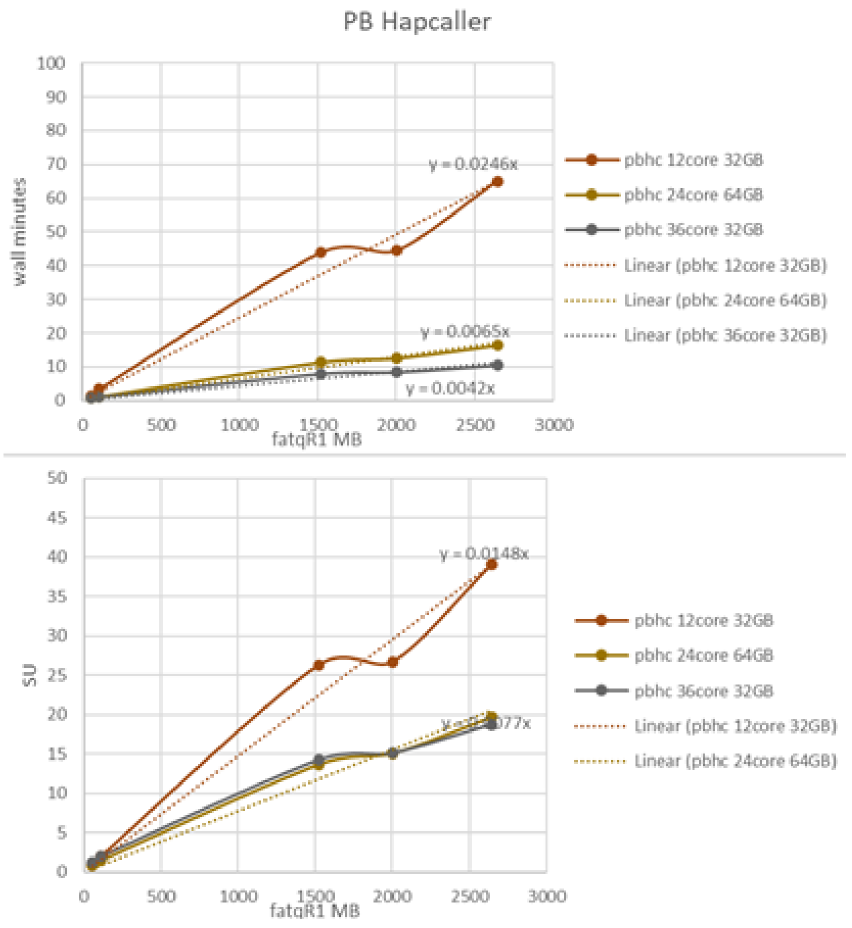


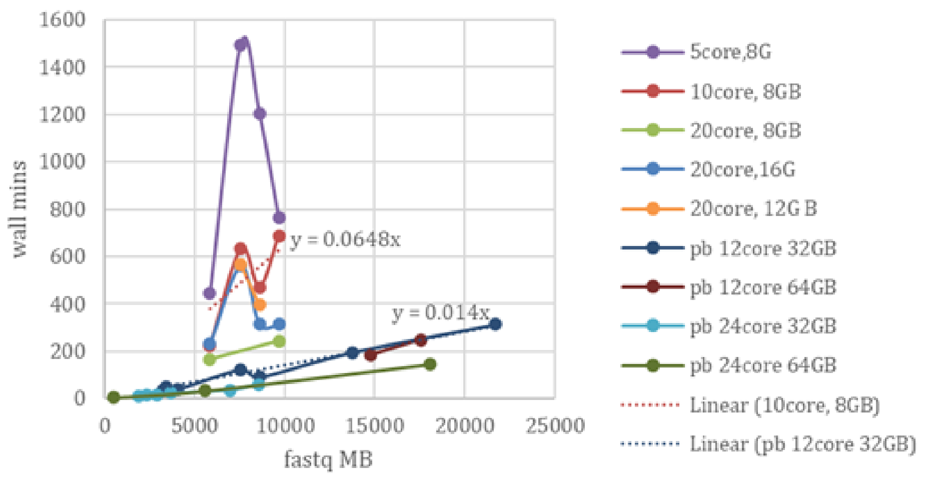
Figure 10. Parabricks haplotypecaller workflow (Figure 4) walltime and service units consumption for different numbers of cores and assigned RAM.

Based on these results we chose the following settings for Parabricks fq2bam and haplotypecaller:

- **Ngpu = 2,**
- **Ncpu = 24,**
- **Mem = 32GB**

Estimation of service units and walltime to process entire dataset

Using the optimal cores and memory determined above, the run time and service unit consumption required to process the entire dataset were estimated for both GATK and Parabricks using the slopes in the figures below and the total size of all input files.



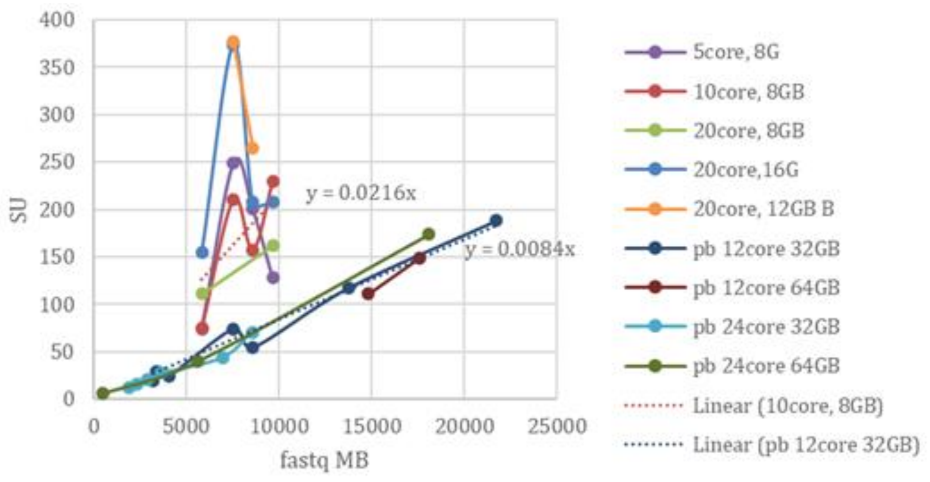
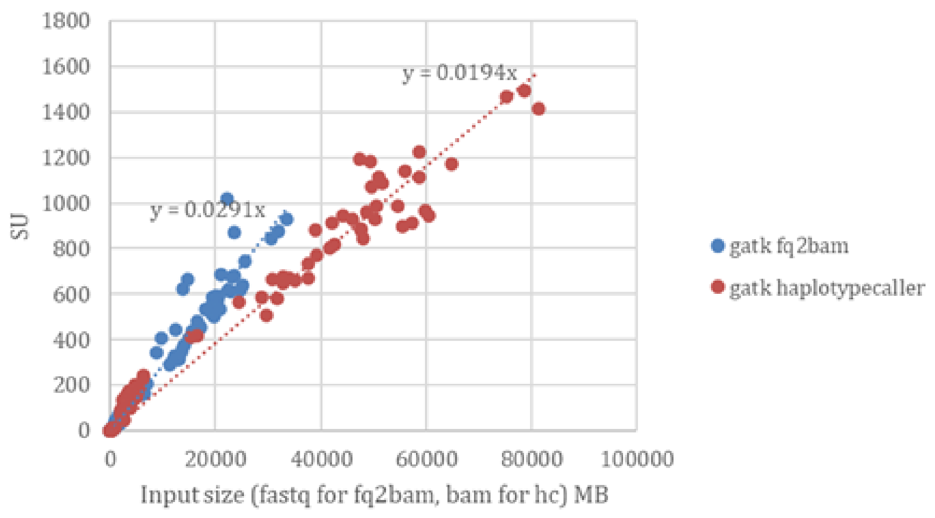


Figure 11. Extrapolation charts to estimate wall time and service units for GATK and PB fastq2bam workflows (figures 1 and 2) using the Fastq input sizes.

The actual service units used to process the entire set of test data, using GATK and Parabricks, are in the figure below.



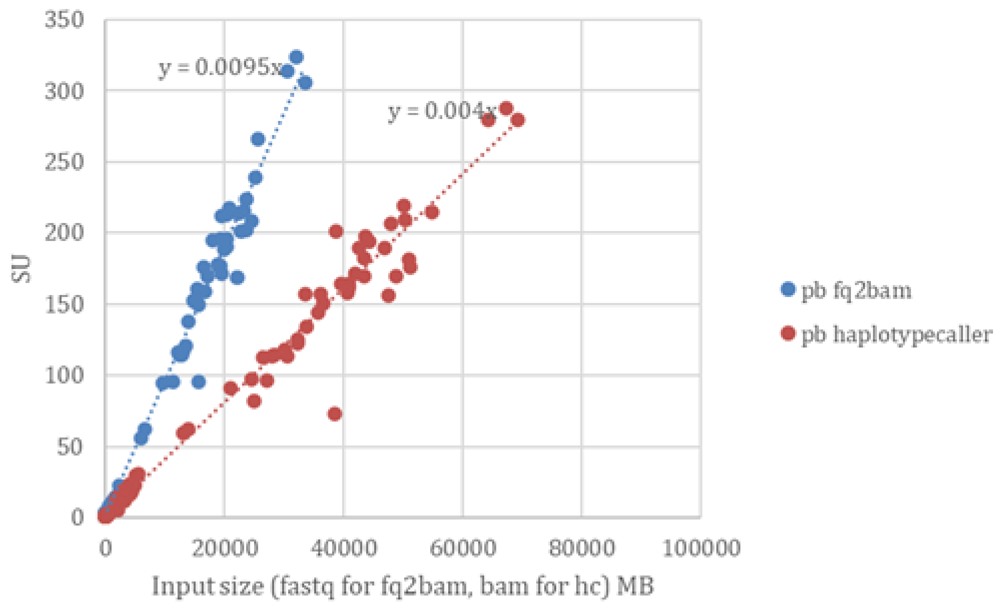


Figure 12. Actual wall time and service units used by GATK (top) and Parabricks for different Fastq input sizes.

For fq2bam:

- GATK estimated SU/fastqMB is 0.0216, actual is 0.029;
- Parabricks estimated SU/fastqMB is 0.0084, actual is 0.0095.

For haplotype caller, still recomputing because estimate was based on fastqMB while actual on bamMB.

Merging of GVCFs and joint genotyping

Downstream processing requires merging of all sample gvcfs into a single gvcf for joint genotyping. GATK uses `genomicsdbimport` and `selectvariants`, while the Parabricks equivalent is being discontinued and NVIDIA recommend the use of glnexus for this step. The processing occurs per chromosome, each point in the figure is for one chromosome gvcf.

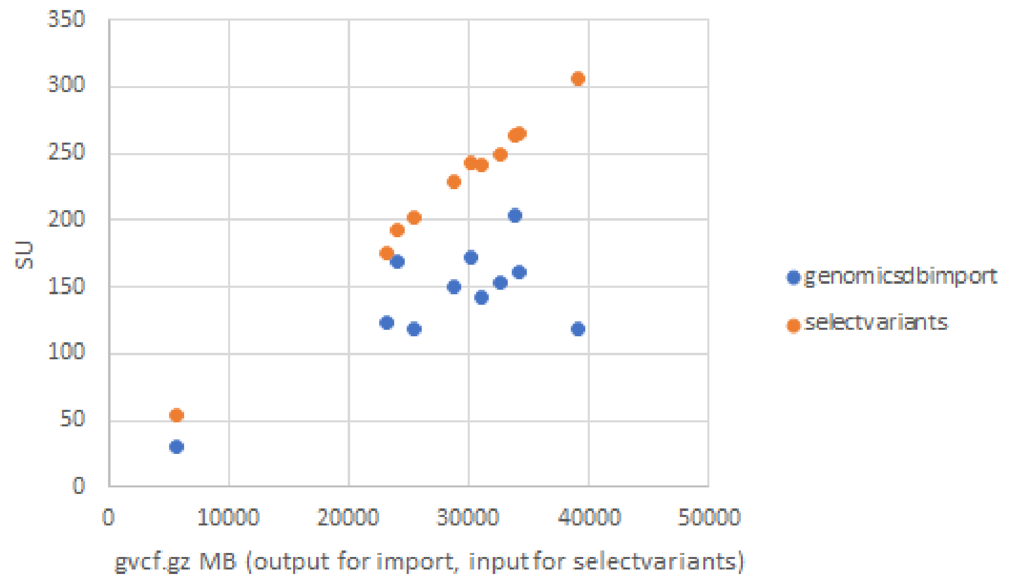


Figure 13. Service units used by GATK GenomicsDBImport and SelectVariants for merging of different input GVCF file sizes.

Then `GenotypeGVCFs` is run for joint genotyping to get the final vcf file. GATK and Parabricks service units in this figure.

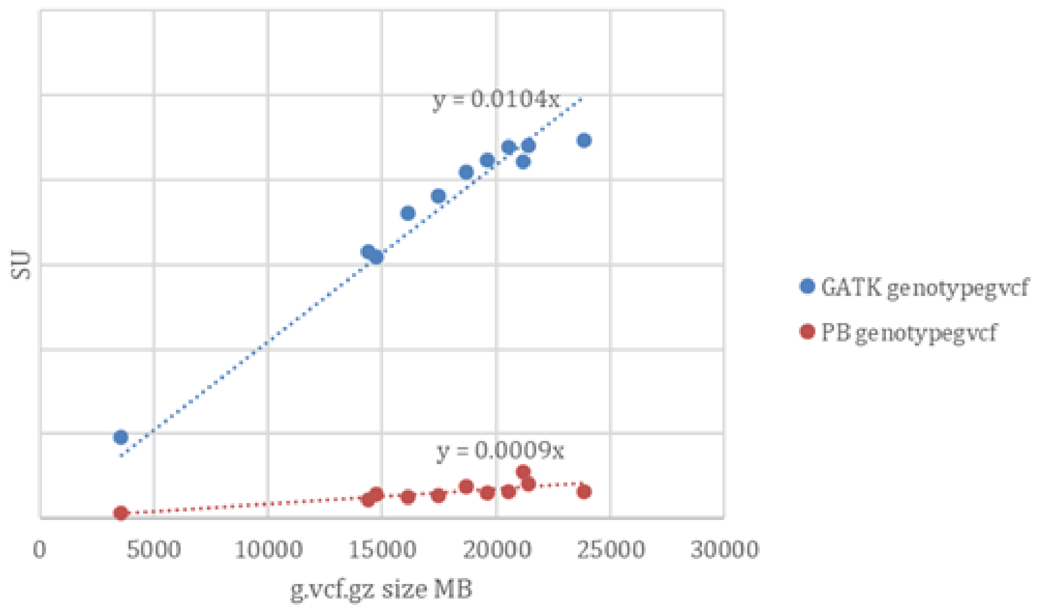


Figure 14. Service units used by GATK and PB GenotypeGVCF for different input GVCF file sizes.

Service units and walltime estimation

The service units requirement to run fastq2bam and haplotypecaller can be estimated using the average fastq file sizes in MB:

avgFastqMB= (fastq_R1.gz + fastq_R2.gz)/2 for pair-end reads, or
fastq.gz/2 for interleaved fastq.gz file

- Using 10 cores for GATK fq2bam (bwa mem + samtools view + gatk AddOrReplaceGroup + gatk SortSam + gatk MarkDuplicatesSpark)
- GATK fq2bam Service Units = 0.0188*avgFastqMB

- Using GATK HaplotypeCaller per chromosome
- GATK haplotypecaller Service Units = 0.0609*avgFastqMB

- Using 2 GPUs, 24 CPUs, 32GB RAM for parabricks
- Parabricks fq2bam Service Units = 0.0084*avgFastqMB

- Parabricks haplotypecaller Service Units = 0.0077*avgFastqMB

Estimating walltime depends on the number of cores used. For the same settings as above:

- GATK fq2bam Walltime mins = 0.0563 * avgFastqMB
- GATK haplotypecaller Walltime mins = 0.1662 * avgFastqMB

- Parabricks fq2bam Walltime mins = 0.0057 * avgFastqMB
- Parabricks haplotypecaller Walltime mins = 0.0042 * avgFastqMB

Based on these values, at the optimal settings identified, the Parabricks germline pipeline is estimated to finish 15 - 20x faster, and use 5x less service units than vanilla GATK.

Benchmarking scripts

Running GATK

The scripts to run GATK for this benchmarking tests on Gadi are available here: https://github.com/Southern-Cross-Plant-Science/GATK-Parabricks_benchmarking_Gadi_NCI/tree/main/GATK_Gadi_Pipeline. The details are explained in the repository and will be summarised here. The pipeline involves multiple scripts to be executed in the order listed below.

Table 2. List of scripts to run GATK on Gadi at NCI.

submit_array_gatkbam.sh	Iterates through all samples, generates a BAM alignment for each FASTQ pairs
submit_array_gatk_hc_byint.sh	Performs variant calling to generates a genomic VCF file g.vcf.gz for each sample and chromosome
submit_array_gatk_genomicsdb_byint.sh	Loads the g.vcf.gz to a GenomicsDB database, one database for each chromosome
submit_array_gatk_genotype_byint.sh	GenotypeGVCF performs the joint genotyping to generate vcf.gz file foreach chromosome
qsub merge_vcfs.sh	Concatenates all the chromosomes vcf.gz into one genome-wide VCF file

These files need to be prepared before running the scripts.

1. paths.sh – located in the same directory as the scripts, paths.sh is called by the other scripts to define variables and file locations.

Table 3. List of variables needed for running GATK at Gadi

JOB_NAME	job name
REFERENCE_DIR	directory where the reference FASTA is stored
REF	reference genome, located in ${REFERENCE_DIR}${REF}.fna

SAMPLE_FILE	file location of samples file
CHROMOSOME_FILE	File location of the chromosomes file

2. SAMPLE_FILE – the sample file lists the tab separated read group id, sample name and file paths (one line per sample).

```
read_group_i sample_name_i  path/to/fastq_i_R1  path/to/fastq_i_R2
```

read_group_i is used for @RGID,@RGLB,@RGPU and sample_name_i for @RGSM in the generated BAM file.

For samples with multiple FASTQ pairs, `sample_name` may be repeated for multiple read groups. Then merge their BAMs manually using picard MergeSAMFiles into a single BAM for the sample.

3. CHROMOSOME_FILE - the chromosome file lists the identifiers of all assembled chromosomes in the FASTA. All other contigs/scaffolds in the FASTA are combined in one VCF. The filename should end with ".intervals".

A directory with name JOB_NAME is created with all the intermediate and result files.

Running Parabricks

The Parabricks pipeline scripts are available here: https://github.com/Southern-Cross-Plant-Science/GATK-Parabricks_benchmarking_Gadi_NCI/tree/main/Parabricks_Gadi_Pipeline. The same files and definitions are required as for the GATK pipeline above, except for NGPU, which is the number of GPUs to use. THREADS should be set at 12x NGPU based on the Gadi specification for gpuvolta queues. The pipeline consists of these scripts to be executed in the given order.

Table 4. List of scripts to run the Parabricks pipeline in Gadi at NCI.

submit_array_pb.sh	Iterates through all samples, generates a BAM alignment for each FASTQ pairs.
submit_array_pb_glnexus_byint.sh	Using GLNEXUS, performs variant calling and joint genotyping to generate a vcf.gz file for each chromosome.
qsub merge_vcfs.sh	Concatenates all the chromosomes vcf.gz into one genome-wide VCF file.

Collecting run statistics

The jobs above generate stdout and stderr reports with file names JOB_NAME.$i.stdout and JOB_NAME.$i.stderr, respectively. $i is the row number of the ith sample defined in the ith line of SAMPLE_FILE.

The final BAM files have the name ${JOB_NAME}-${group_id}-${BWA}-rg-dup.bam, and the GVCF names are ${JOB_NAME}-${group_id}-${BWA}.g.vcf.gf

python3 summarize-stdout.py JOB_NAME BWA OUTFILE REF

This script collects and tabulates the runtime information for all subjobs submitted through submit_array.sh. It should be called from the same directory where submit_array.sh was run. This generates the file summary_${JOB_NAME}.txt

BWA parameter can be: bwa1 for GATK, fq2bam for parabricks
REF is reference genome as used in submit_array.sh, defaults to cs10

OUTFILE is the workflow to collect statistics. The possible values are in this table:

Table 5. Values for OUTFILE

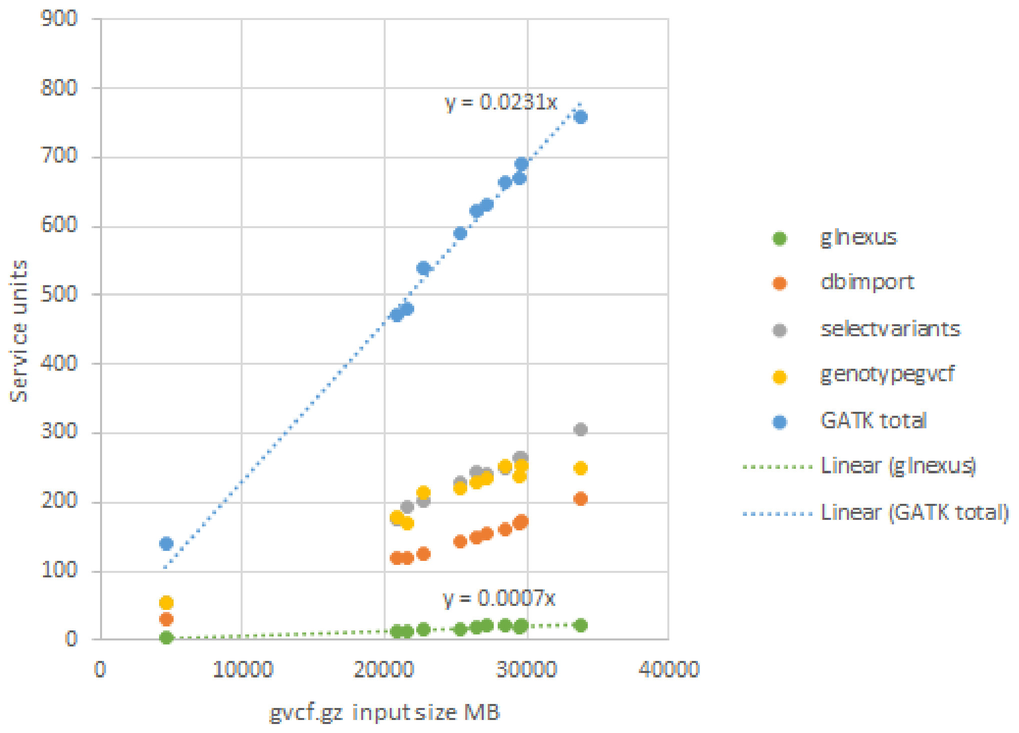
dup	GATK fastq to bam
hc	GATK gvcf
pbdup	Parabricks fastq2bam
pbhc	Parabricks haplotypecaller

The summary table has these columns:

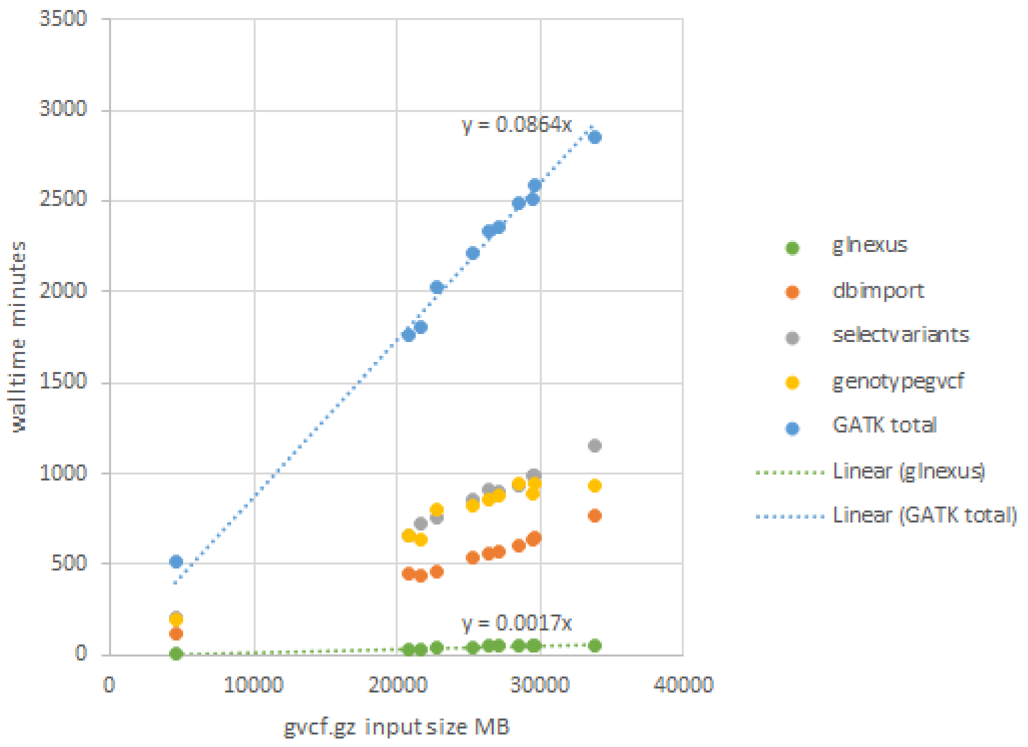
```
READ                sample name
FASTQ1_MB           file size
FASTQ2_MB           file size
SU                  service units
EXIT                exit status
BAMSIZE_MB          generated BAM file size
OUTSIZE_MB          generated GVCF file (or BAM for OUTFILE of dup or pbdup)
WALLTIME            wall time
MEMORY              memory used MB
NCPU                cpu used
JOBFS               jobfs sized used MB
JOBID               jobid
CPUTIME             CPU time
%CPU                CPU TIME / (N * WALLTIME)
WALL_MINS           wall time in minutes
BWA_MINS            fastq2bam cpu time in mins
HC_MINS             haplotypecaller cpu time in mins
N_FASTQ             number of fastq files
AVG_FASTQ           avg fastq file size MB
FQ2BAM_SEC          fastq2bam cpu time in secs
HC_SEC              haplotypecaller cpu time in secs
```

Joint genotyping GATK (importdb+selectvariants+genotypegvcf+concat) vs Parabricks glnexus

GVCF files were joint-genotyped using GATK and Parabricks glnexus to produce the final VCF file. GATK was run per chromosome since the job takes longer than 48 hours if using the entire chromosome. For glnexus, both per chromosome and whole genome runs were performed.The charts below compare the service units, CPU time and wall time for the per chromosome runs for both glnexus and GATK, with the contributions for each GATK step broken down further. The variant positions for the per chromosome and whole genome using glnexus perfectly matched.



Service units
gvcf.gz input size MB
y = 0.0231x
y = 0.0007x
glnexus
dbimport
selectvariants
genotypegvcf
GATK total
Linear (glnexus)
Linear (GATK total)



walltime minutes
gvcf.gz input size MB
y = 0.0864x
y = 0.0017x
glnexus
dbimport
selectvariants
genotypegvcf
GATK total
Linear (glnexus)
Linear (GATK total)

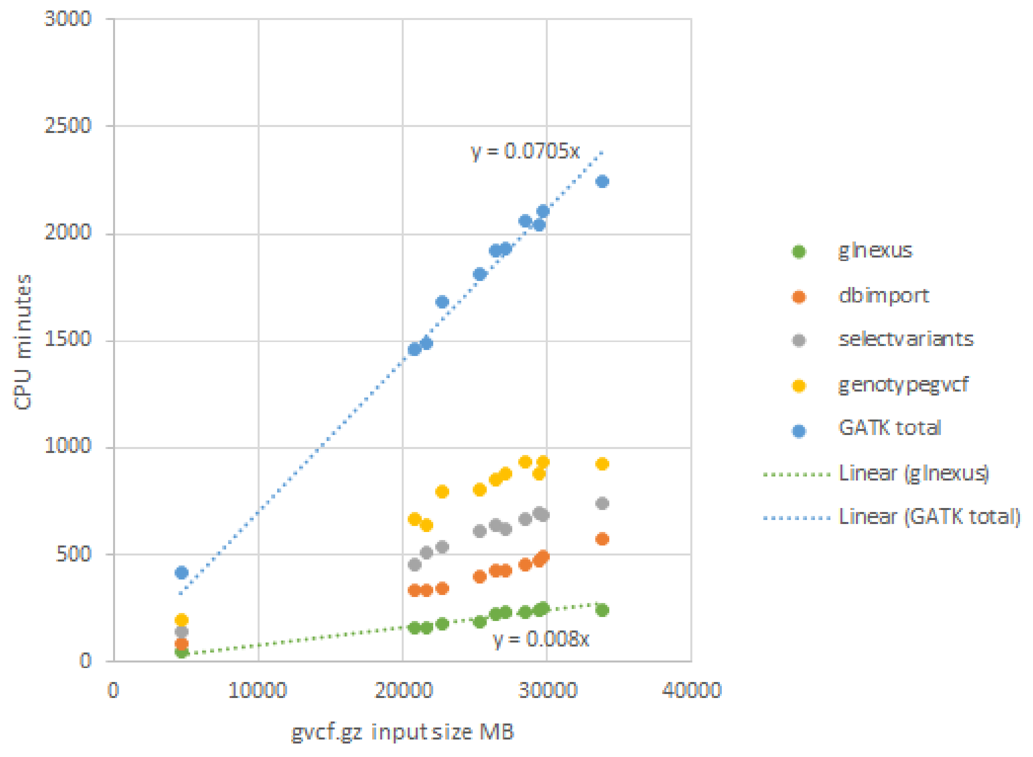


Figure 15. Service units, wall time and CPU time for joint genotyping using Parabricks GLNEXUS and GATK workflow.

Compare results for the different methods

The genotyping pipeline can be separated into two stages, variant calling and joint genotyping. Variant calling consists of alignment and haplotype calling. The methods used for each steps are:

Table 6. Tools tested for the GATK and Parabricks genotyping pipelines.

Step	Pipeline (code)	Tools
Variant calling	GATK bwa1	bwa mem + sort sam + markduplicate + haplotypecaller
	Parabricks (fq2bam)	fq2bam + haplotypecaller
Joint genotyping	GATK (genotypegvcf)	dbimport + selectvariants + genotypeGVCFs
	Parabricks (glnexus)	glnexus
	Parabricks (pbgenotypegvcf)	deprecated pbrun genotypegvcf (requires a combined gvcf, tools to combine are not available in parabricks and had to use dbimport + selectvariants from vanilla GATK)

To quantify the effects of the pipeline used, four sets of vcf were generated using different combinations:

1. GATK-GATK genotypegvcf
2. GATK-glnexus
3. PB-GATK genotypegvcf
4. PB-glnexus

Effect of joint genotyping, with GVCF inputs generated by GATK and Parabricks.

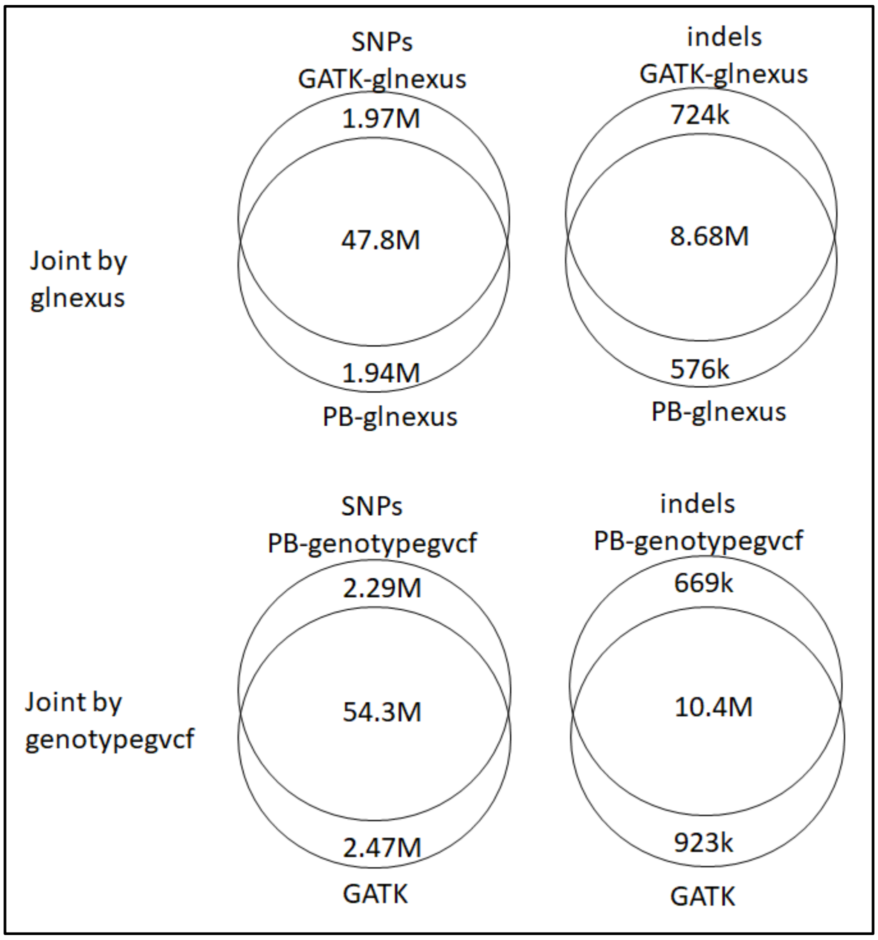


Figure 16. Number of variants discovered and shared between usingGLNEXUS and GenotypeGVCF for joint genotyping.

Effect of haplotypecaller, with joint genotyping using GATK genotypegvcf or glnexus.

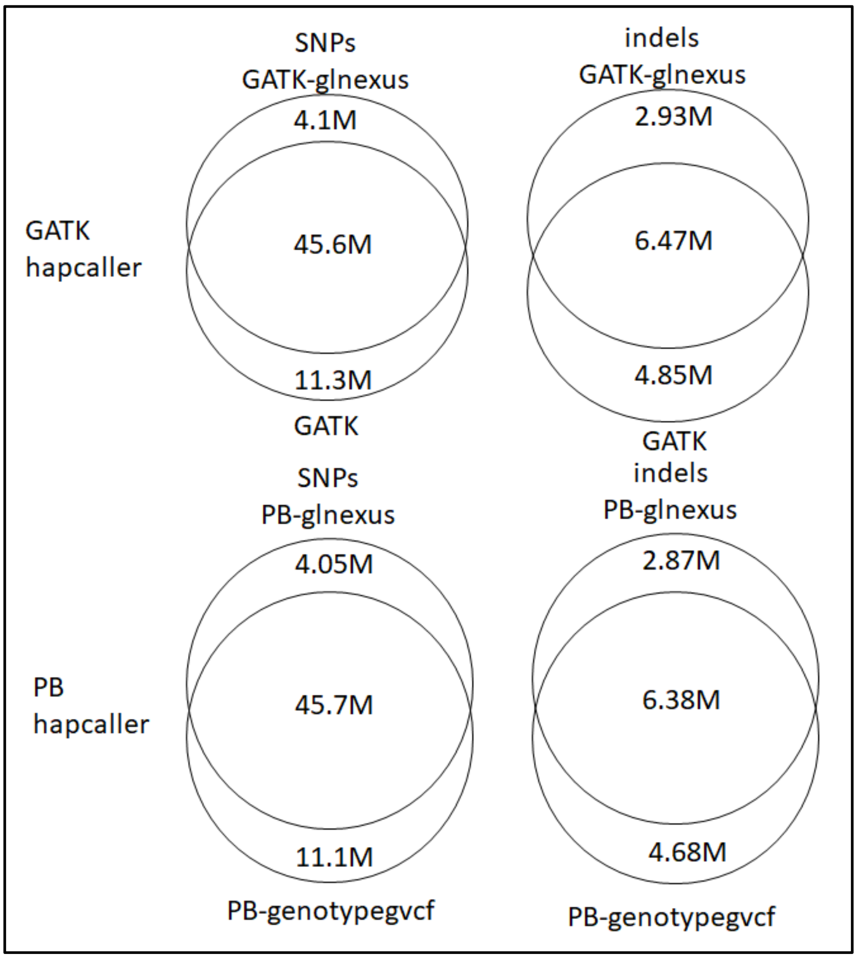


Figure 17. Number of variants discovered and shared between using GATK and PB haplotypecaller.

Using GATK-genotypegvcf vs Parabricks-glnexus.

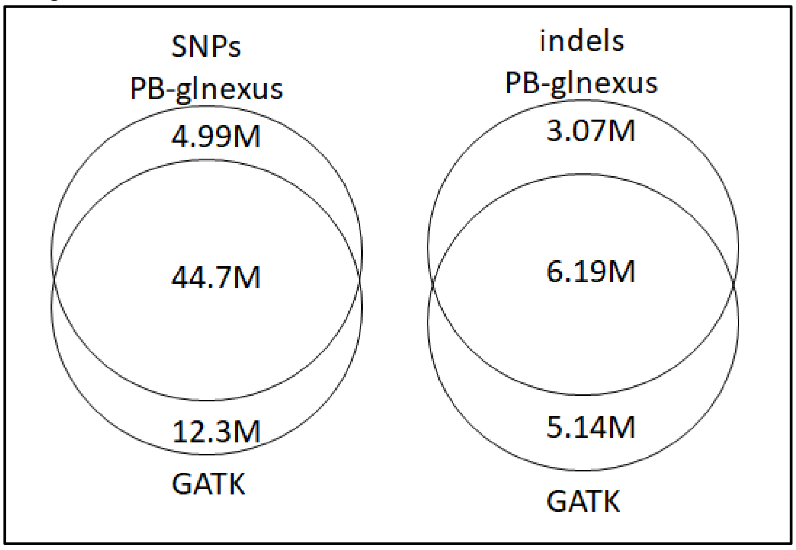


Figure 18. Number of variants discovered and shared between GATK-genotypegvcf and Parabricks-glnexus.

The deprecated Parabricks pbrun genotypegvfc (pbgenotypegvfc) was also tested using the combined gvcf generated by vanilla GATK (dbimport + selectvariants + bcftools concat).. The result shows it is equivalent to the vanilla GATK GenotypeGVCFs (genotypegvcf).

PB-pbgenotypegvcf = PB-genotypegvcf
GATK-pbgenotypegvcf = GATK-genotypegvcf

The effect of haplotype caller using genotypegvfc for joint genotyping.

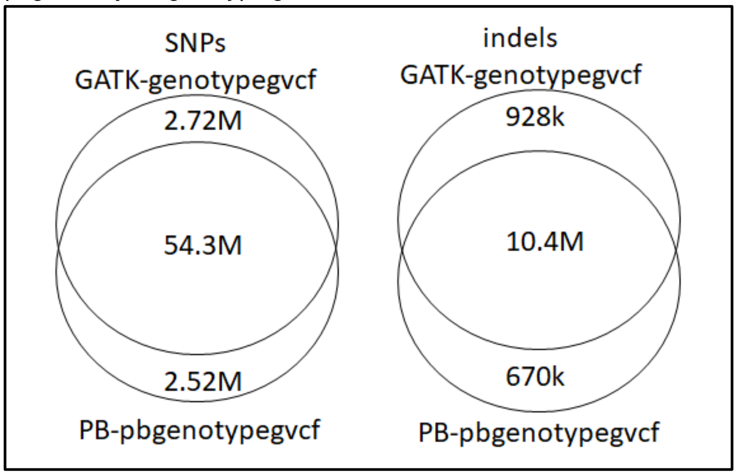


Figure 19. Number of variants discovered and shared between GATK GenotypeGVCF and PB genotypegvcf.

The above comparisons show that glnexus contributed to most of the difference between the vanilla GATK and Parabricks results. Vanilla GATK discovered more variants (57 M SNPS, 11.3 M indels) compared to Parabricks-glnexus (49.7 M SNPs, 9.26M indels). However, the Parabricks version of genotypegvcf gives the same results with the vanilla GATK GenotypeGVCFs, and the older Parabricks pipeline discovered (56.8 M SNPS, 11 M indels).

Parabricks-glnexus service units for different reference genome size

Parabricks was also used to call variants using other reference genomes with different sizes.

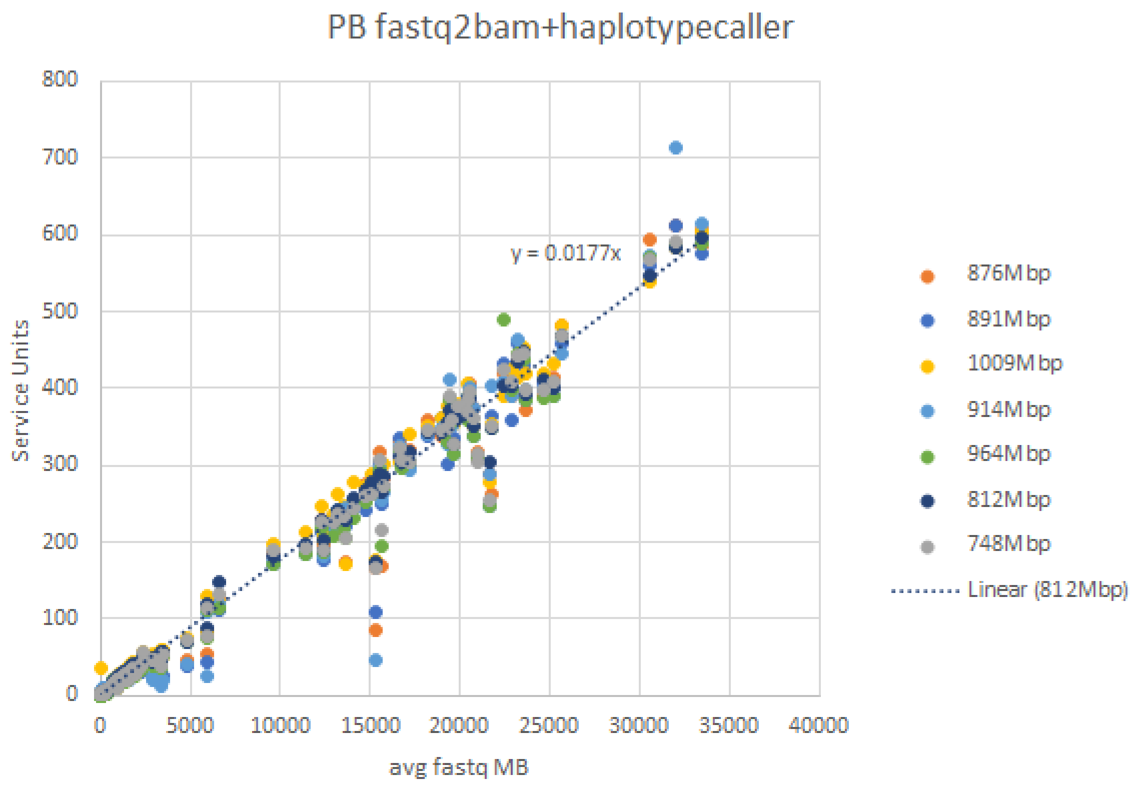


Figure 20. Service units used for Parabricks fastq2bam + haplotypecaller for different fastq and reference genome assemblies.

The number of cores and gpus used were found to be optimal (48 cores, 4 gpus) for fq2bam. Increasing them did not improve the wall time, and only increased the service units consumed. The same result was observed with glnexus: 5 cpu cores was optimal and increasing these further did not improve wall time.

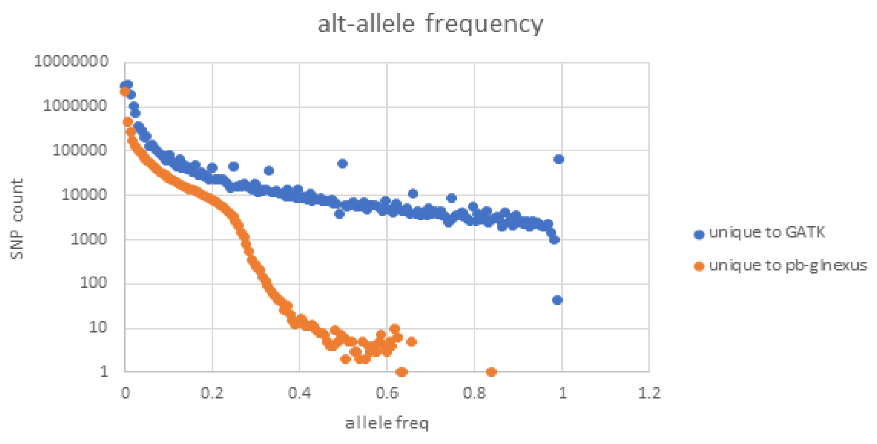
Rare allele discovery rate

It would be difficult to assess accuracy because we do not have a gold standard, or experimentally validated variants, for our species. Different tools may be compared using other criteria like the discovery rate, especially discovery of rare alleles. The discovery rates for *C. sativa* with different pipelines are shown in the table below. *O. sativa* is included for comparison although it used ~19x more samples.

Table 7. Comparison of variant discovery rates between pipelines and with other species

Genome	Size, bp	Samples	Pipeline	SNPs / 10kb
C. sativa, CBDRx cs10	876,147,649	162	GATK	651
			PB-genotypegvcf	630
			PB-glnexus	568
O. sativa, Nipponbare	373,245,519	3024	GATK	859

The alternate allele frequencies of the unique SNPs between GATK, PB-pbgenotypegvcf and PB-glnexus are plotted in the figure below. It shows that the Parabricks fq2gvcf is comparable with vanilla GATK in discovering alleles, but the joint genotyping between using genotypegvcf or glnexus made the difference. Both vanilla GenotypeGVCF and the deprecated pbgenotypegvcf discovered more rare alleles (the regions approaching AF=0.0 and AF=1.0 in the graph) than glnexus during joint genotyping.



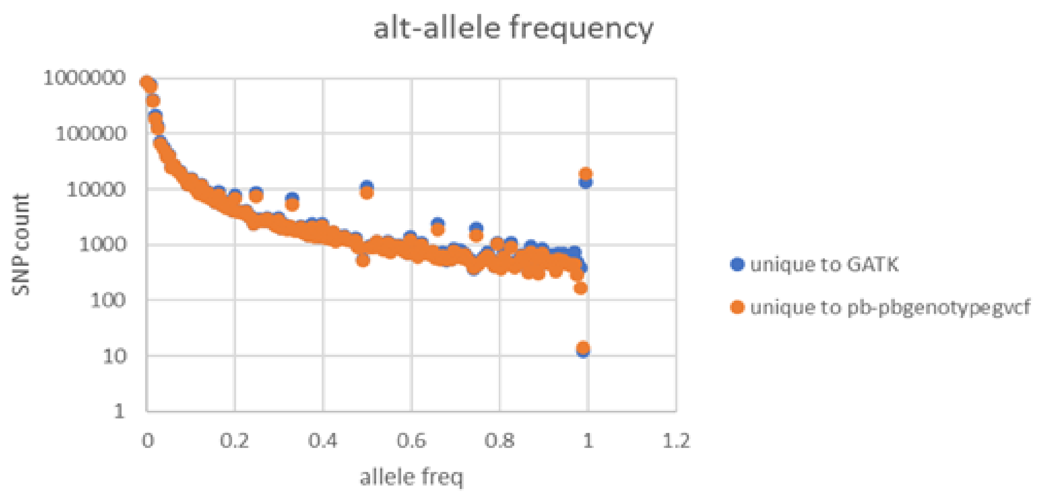


Figure 21. Comparison of discovery rate for different Alternate Allele Frequencies between GATK and PB-GLNEXUS (top) and between GATK and PB_GenotypeGVCF.

Allele frequencies for all SNPs of the three pipelines.

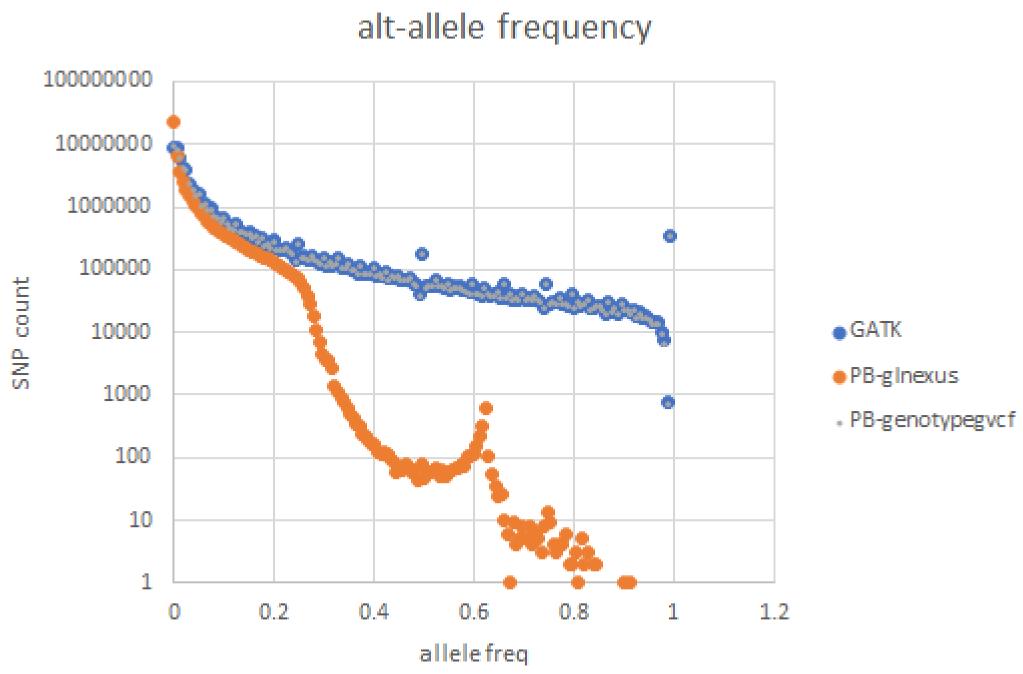


Figure 22. Discovery rate for different Alternate Allele Frequencies for the three tested pipelines.

RNA-Seq Variant Discovery

The GATK RNA-Seq short variant discovery pipeline at https://gatk.broadinstitute.org/hc/en-us/articles/360035531192-RNAseq-short-variant-discovery-SNPs-Indels- uses the splice-aware aligner STAR (https://github.com/alexdobin/STAR), to align transcript reads into the genome. Alignments that span introns are then split into supplementary alignments and mismatching overhangs are clipped. RNA aligner conventions and mapping qualities are also remapped to match DNA conventions. After these pre-processing steps, the resulting BAM file is used for variant calling similar to DNA read alignment BAMs in the previous sections.

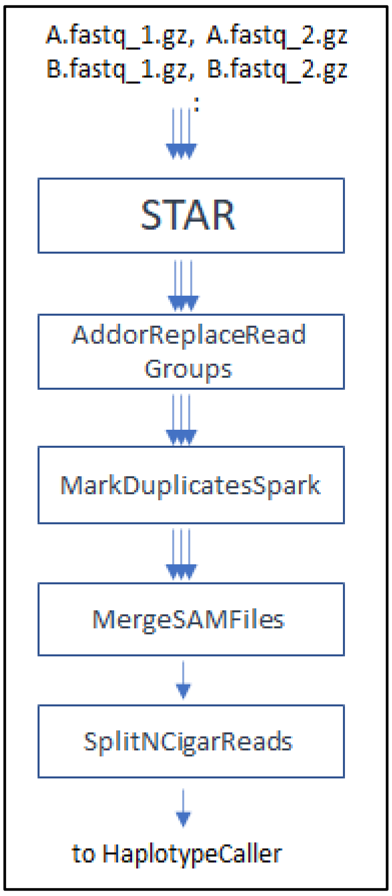


Figure 23. GATK RNA-Seq data alignment workflow, rna_fq2bam

Here we compare the vanilla GATK RNA-Seq preprocessing and the Parabricks rna_fq2bam pipeline (https://docs.nvidia.com/clara/parabricks/v3.5/text/rna.html#rna-fq2bam) plus PB splitncgar tool (https://docs.nvidia.com/clara/parabricks/v3.5/text/qc_metrics.html#splitncigar).

For the variant calling step using GATK HaplotypeCaller or PB haplotypecaller, the comparisons described in the previous sections also apply. We are using 17 RNA-Seq samples with 3 replicates. Initially, 21 samples were available but Parabricks rna_fq2bam did not finish after 48 hrs on 4 samples, so only 17 were used to proceed to haplotype calling and comparison.

The service units consumed as a function of input size for each tool in the pipeline for both vanilla GATK and Parabricks are plotted in the figure below. Walltime was not compared because most of the vanilla tools (STAR, MarkDuplicateSpark, SplitNCigarReads) support multi-threading. 24 cores were used for both pipelines, plus 2 GPUs for Parabricks. The result shows the vanilla pipeline consumed five times more service units than Parabricks.

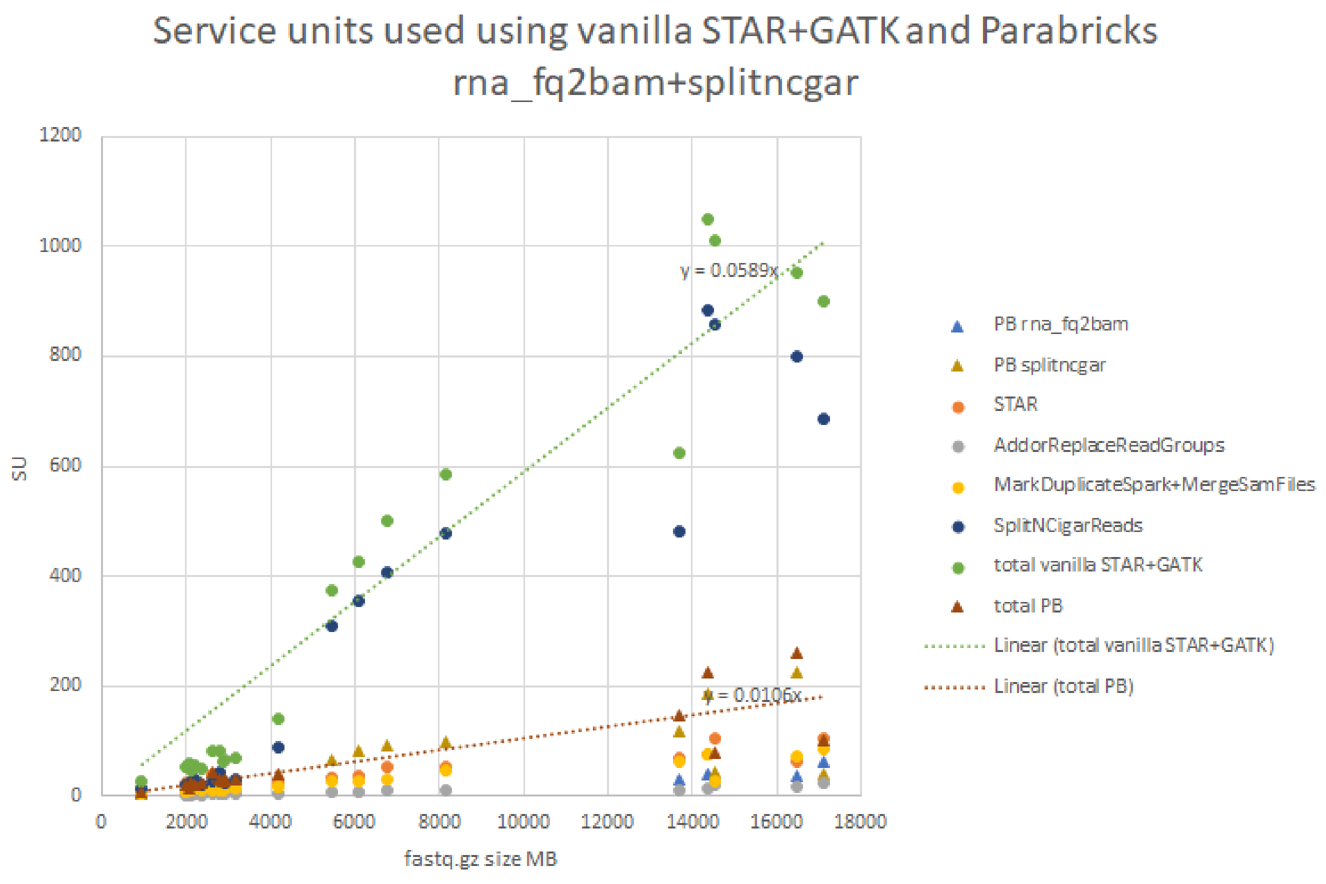


Figure 24. Service units usage for GATK-RNA-Seq and Parabricks pipelines for different fastq input sizes.

There is a large discrepancy on the number of aligning reads in the resulting BAM files between the two pipelines. In the figure below, samtools (samtools view -c) was used to get the number of aligning reads in the BAM file for each sample. The number of reads that are aligned using GATK pipeline with vanilla STAR are more than twice that of using Parabricks rna_fq2bam+splitncgar. Except for the results to output, both '--two-pass-mode Basic' and default settings were used. The commands can be checked in the github scripts. This information alone indicates that the vanilla pipeline has apparently discovered more variants than the Parabricks pipeline.

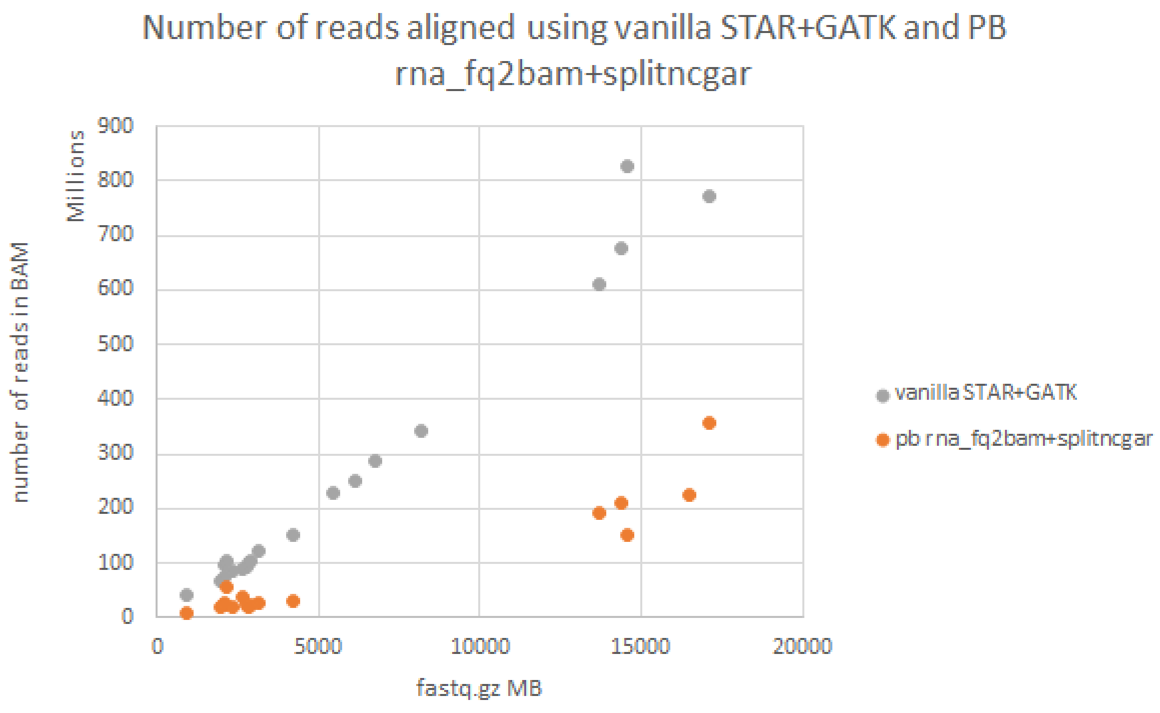


Figure 25. Comparison of the number of RNA reads that are aligned for GATK and Parabricks RNASeq Variant discovery pipelines.

Comparison of RNA-Seq variants discovery

Haplotype caller and joint genotyping were performed on both BAM files using the same method (PB genotypegvcf) to determine the difference in variants discovered. The number of common and unique variants discovered using vanilla GATK pipeline and Parabricks are compared in the Venn diagram below.

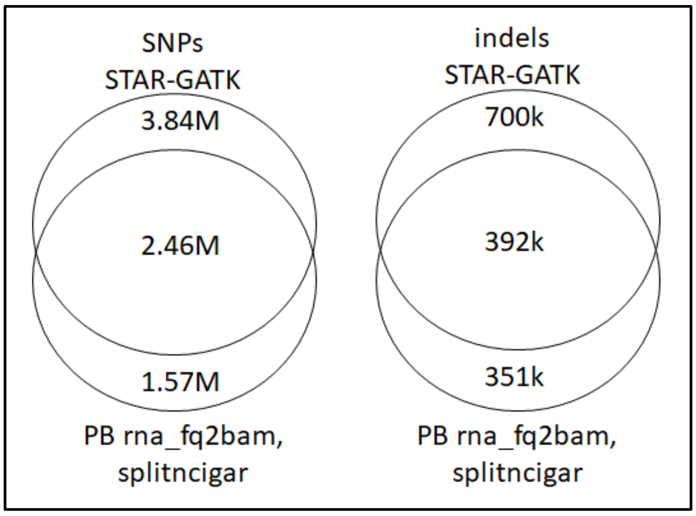


Figure 26. Number of variants discovered and shared between GATK and Parabricks RNASeq Variant discovery pipelines.

The result shows the vanilla pipeline discovered more SNPs (6.3 M vs 4.03 M) and indels (1.09 M vs 743 k) than the parabricks pipeline.

STAR + GATK pipeline

The scripts to run the RNA-Seq variant-calling are included in the GATK pipeline, RNA-Seq sequence variant-calling section. The pipeline involves multiple scripts to be executed in the order listed below.

Table 8. List of scripts to run variant calling using RNA-Seq data.

submit_array_gatk_rna_fq2bam.sh	Iterates through all samples, generates a BAM alignment for each sample using triplicate FASTQ pairs
submit_array_gatk_rna_hc_byint.sh	Performs variant calling to generates a g.vcf.gz for each sample and chromosome
submit_array_rna_genomicdb_byint.sh	Loads the g.vcf.gz to a GenomicsDB database and performs joint genotyping to generate a vcf.gz file for each chromosome
qsub merge_rna_vcfs.sh	Concatenates all the chromosomes vcf.gz into one genome-wide VCF file

These files need to be prepared before running the scripts.

1. paths_rna.sh – located in the same directory as the scripts, paths.sh is called by the other scripts to define variables and file locations.

Table 9. List of variables to define in path.sh for running STAR-GATK RNA-Seq variant discovery in Gadi

JOB_NAME	job name
REFERENCE_DIR	directory where the reference FASTA is stored
REF	reference genome, located in ${REFERENCE_DIR}${REF}.fna
SAMPLE_FILE	file location of samples file
CHROMOSOME_FILE	File location of the chromosomes file
STAR_PATH	path to STAR executable
STAR_GENOMEDIR	STAR reference genome directory, generated using STAR --runMode genomeGenerate

2. SAMPLE_FILE – the sample file lists the tab separated sample name, read groups and file paths, one line per sample.

```
SMi  RGi1 path/to/FQi1_R1 path/to/FQi1_R2  RGi2 path/to/FQi2_R1 path/to/FQi2_R2 RGi3 path/to/RGi3_1 path/to/RGi3_2
```

3. CHROMOSOME_FILE - the chromosome file lists the identifiers of all assembled chromosomes in the FASTA. All other contigs/scaffolds in the FASTA are combined in one VCF. The filename should end with ".intervals".

A directory with name JOB_NAME is created with all the intermediate and result files.

Parabricks commands

The Parabricks pipeline scripts for RNA-Seq variant calling are included in the repository. The same files and definitions are required as for the GATK pipeline above, except for NGPU which is the number of GPUs to use. THREADS should be set at 12 x NGPU based on Gadi specification for gpuvolta queues. The pipeline consists of these scripts to be executed in the given order.

Table 10 List of scripts to run the Parabricks RNA-Seq variant discovery pipeline in Gadi at NCI.

submit_array_pb_rna.sh	Iterates through all samples, generates a BAM alignment for each sample
submit_array_pb_rna_glnexus_byint.sh	Using GLNEXUS, performs variant calling and joint genotyping to generate a vcf.gz file for each chromosome
qsub merge_vcfs_rna.sh	Concatenates all the chromosomes vcf.gz into one genome-wide VCF file

Variant calling on single sample

To isolate the effects of joint genotyping on the results, we checked the concordance when using only one sample. GATK GenotypeGVCFs was run on the g.vcf.gz files already generated before from the HaplotypeCaller of both GATK and Parabricks pipelines. We presume that having only one sample, GenotypeGVCFs has no further basis in altering the variants, and will just remove the GVCFBlock and <NON_REF> in the gvcf, to generate the VCF file. Haplotype caller could have been used to generate the vcf directly without passing to g.vcf.

vcf-compare (https://vcftools.github.io/perl_module.html#vcf-compare) was used in the comparison of vcf files.

Variant calling on genomic sequence

For genomic variant calling, two samples, SRR3294430.1, SRR3294441.1 (both Afghan Kush) using the cs10 reference assembly (GCF_900626175.2) were tested individually. For SRR3294441.1, comparison of unique and common sites gives a high concordance:

- Unique to GATK 97651 (2.1%),
- Unique to PB: 75545 (1.6%),
- Common 4509620 (97.9% of GATK, 98.4% of PB)

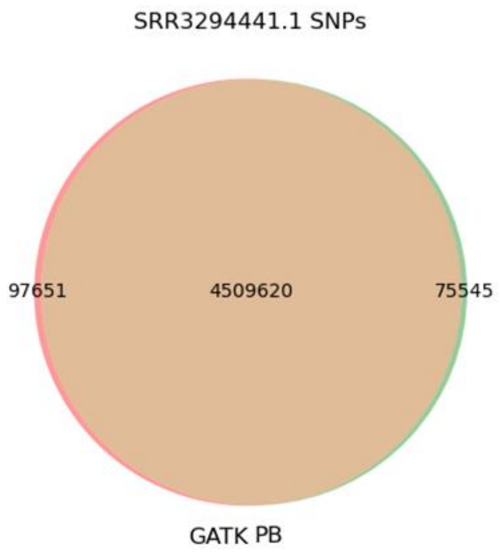
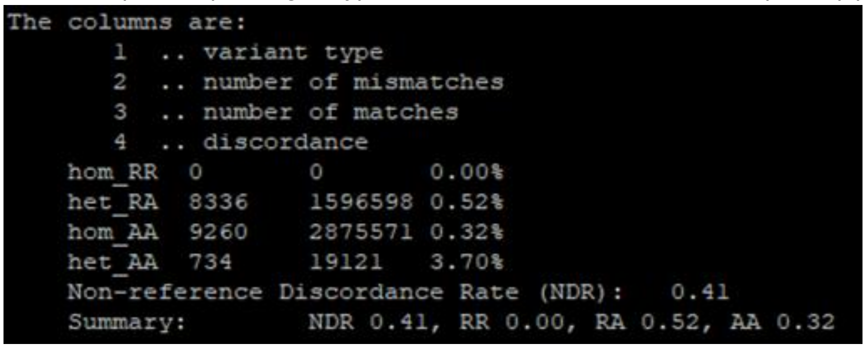


Figure 27. Number of unique and common SNP positions between GATK and Parabricks pipelines using only one sample.

Genotype comparison on common sites gives a non-reference discordance rate (NDR) of 0.41% (mismatches over total calls involving non-reference alleles).

Table 11. Vcf-compare output for genotype concordance between GATK and fq2bam pipelines.



```
The columns are:
       1  .. variant type
       2  .. number of mismatches
       3  .. number of matches
       4  .. discordance
    hom_RR  0       0       0.00%
    het_RA  8336    1596598 0.52%
    hom_AA  9260    2875571 0.32%
    het_AA  734     19121   3.70%
    Non-reference Discordance Rate (NDR):   0.41
    Summary:        NDR 0.41, RR 0.00, RA 0.52, AA 0.32
```

The same comparison for SRR3294430.1 gives:

- Unique to GATK 126236 (2.5%),
- Unique to PB: 99086 (2.0%),
- Common 4840704 (97.5% of GATK, 98.0% of PB), and
- NDR of 0.53%

Variant calling on RNA-Seq sequence

For RNA-Seq variant calling, STAR-GATK discovered more variant sites compared to Parabricks using the rna_fq2bam pipeline. Using the Afghan Kush RNA-Seq data (SRR10871515 to SRR10871527) on cs10 assembly (GCF_900626175.2), the Venn diagram of discovered sites is shown with Unique to STAR-GATK 259704 (39.1%), unique to PB 41811 (9.4%), common 403662 (90.6% of PB, 60.9% of STAR-GATK). Genotype comparison on common sites gives a non-reference discordance rate of 6.97%.

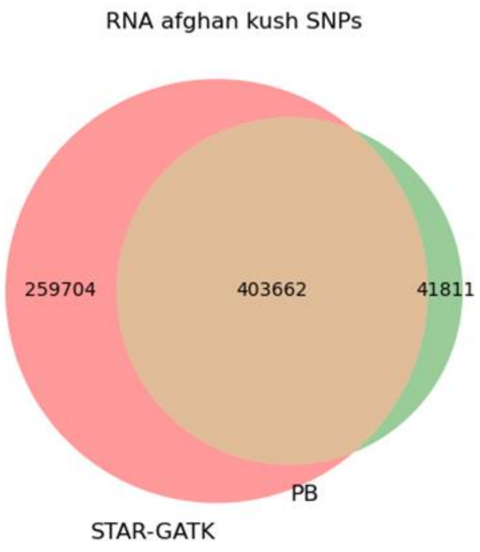
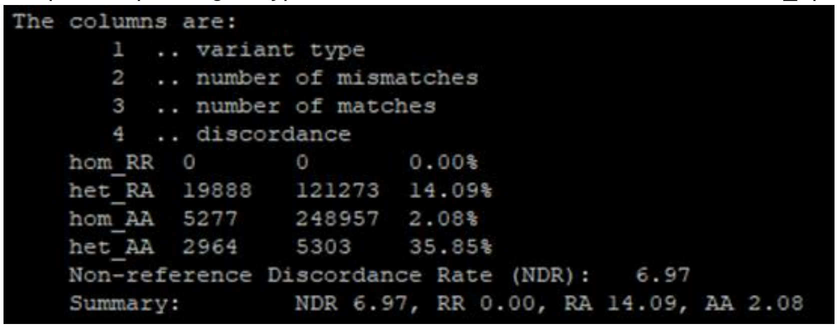


Figure 28. Number of unique and common SNP positions between STAR-GATK and Parabricks pipelines using only one sample for RNA-Seq variant calling.

Table 12. Vcf-compare output for genotype concordance between STAR-GATK and rna_fq2bam pipelines.



```
The columns are:
        1  .. variant type
        2  .. number of mismatches
        3  .. number of matches
        4  .. discordance
    hom_RR  0        0        0.00%
    het_RA  19888    121273   14.09%
    hom_AA  5277     248957   2.08%
    het_AA  2964     5303     35.85%
    Non-reference Discordance Rate (NDR):   6.97
    Summary:         NDR 6.97, RR 0.00, RA 14.09, AA 2.08
```

Another RNA-Seq sample, finola bulbous trichome (SRR7630401, SRR7630403, SRR7630404), was compared. Site comparison gives:

- unique to GATK 160803 (35.8%),
- unique to PB 21479 (6.9%),
- common 288281 (93.1% of PB, 64.2% of STAR-GATK), and
- Genotype comparison gives NDR 16.26%

The above comparisons show fq2bam gives highly concordant results with GATK for DNA/genomic variant calling. However, the rna_fq2bam in Parabricks RNA-Seq variant discovery pipeline behaved similarly for multiple and single samples, discovering less SNPs compared to the legacy STAR-GATK pipeline.

Conclusion

Parabricks enabled fast alignment and variant discovery for large cohorts with low service units consumption. However, the joint genotyping step is difficult to perform without the tool to merge GVCFs, and we needed to use vanilla GATK to do the merging task. The Parabricks implementation of genotypegvcf is comparable in result with the vanilla GATK GenotypeGVCF but is planned to be deprecated. The alternative glnexus is convenient to use and faster but has a lower discovery rate than vanilla GATK.

We recommend keeping the parabricks genotypegvcf and to implement a convenient tool to merge GVCFs as an alternative to GenomicsDB in GATK. A suggestion is for glnexus to also output the merged gvcf in addition to the final bcf, because non-variant information from the gvcfs cannot be recovered from the final bcf nor vcf.

Acknowledgements / citations / credits

The authors acknowledge the provision of computing and data resources provided by the Australian BioCommons Leadership Share (ABLeS) program (Manos et al., 2022). This program is co-funded by Bioplatforms Australia (enabled by NCRIS), the National Computational Infrastructure and Pawsey Supercomputing Research Centre.

The authors also acknowledge the technical support provided by Dr. Johan Gustafsson and Dr. Ziad Al Bkhetan of Australian BioCommons; Brian Davis and Dr. Matthew Downton of the National Computational Infrastructure (NCI); and NVIDIA Technical Support.

https://docs.nvidia.com/clara/parabricks/v3.0/text/software_overview.html

http://samtools.github.io/bcftools/bcftools.html

GATK Germline Pipeline

McKenna, A., Hanna, M., Banks, E., Sivachenko, A., Cibulskis, K., Kernytsky, A., Garimella, K., Altshuler, D., Gabriel, S., Daly, M., & DePristo, M. A. (2010). The Genome Analysis Toolkit: A MapReduce framework for analyzing next-generation DNA sequencing data. *Genome Research*. https://doi.org/10.1101/gr.107524.110.20

Dobin, A., Davis, C. A., Schlesinger, F., Drenkow, J., Zaleski, C., Jha, S., Batut, P., Chaisson, M., & Gingeras, T. R. (2013). STAR: Ultrafast universal RNA-seq aligner. *Bioinformatics*, *29*(1), 15–21. https://doi.org/10.1093/bioinformatics/bts635

Genome assemblies and Illumina reads used are from these publications and public sites

Booth, J. K., Yuen, M. M. S., Jancsik, S., Madilao, L. L., & Page, A. J. E. (2020). Terpene synthases and terpene variation in cannabis sativa1[OPEN]. *Plant Physiology*, *184*(1), 130–147. https://doi.org/10.1104/PP.20.00593

Braich, S., Baillie, R. C., Spangenberg, G. C., & Cogan, N. O. I. (2020). A new and improved genome sequence of Cannabis sativa. *BioRxiv*, 10.5524/100821. https://doi.org/10.1101/2020.12.13.422592

Gao, S., Wang, B., Xie, S., Xu, X., Zhang, J., Pei, L., Yu, Y., Yang, W., & Zhang, Y. (2020). A high-quality reference genome of wild Cannabis sativa. *Horticulture Research*, *7*(1). https://doi.org/10.1038/s41438-020-0295-3

Grassa, C. J., Weiblen, G. D., Wenger, J. P., Dabney, C., Poplawski, S. G., Motley, S. T., Michael, T. P., & Schwartz, C. J. (2021). A new Cannabis genome assembly associates elevated cannabidiol (CBD) with hemp introgressed into marijuana . *New Phytologist*, 10.1111/nph.17243. https://doi.org/10.1111/nph.17243

Kannapedia https://www.medicinalgenomics.com/kannapedia-fastq

Laverty, K. U., Stout, J. M., Sullivan, M. J., Shah, H., Gill, N., Holbrook, L., Deikus, G., Sebra, R., Hughes, T. R., Page, J. E., & Van Bakel, H. (2019). A physical and genetic map of Cannabis sativa identifies extensive rearrangements at the THC/CBD acid synthase loci. Genome Research, 29(1), 146–156. https://doi.org/10.1101/gr.242594.118

Livingston, S. J., Quilichini, T. D., Booth, J. K., Wong, D. C. J., Rensing, K. H., Laflamme-Yonkman, J., Castellarin, S. D., Bohlmann, J., Page, J. E., & Samuels, A. L. (2020). Cannabis glandular trichomes alter morphology and metabolite content during flower maturation. *Plant Journal*, *101*(1), 37–56. https://doi.org/10.1111/tpj.14516

Lynch, R. C., Vergara, D., Tittes, S., White, K., Schwartz, C. J., Gibbs, M. J., Ruthenburg, T. C., deCesare, K., Land, D. P., & Kane, N. C. (2016). Genomic and Chemical Diversity in Cannabis. In *Critical Reviews in Plant Sciences* (Vol. 35, Issues 5–6). https://doi.org/10.1080/07352689.2016.1265363

Manos, S., Gustafsson, O. J. R., Al Bkhetan, Z., & Francis, R. (2022). *Building community data assets for life sciences through ABLeS - the Australian BioCommons Leadership Share*. Zenodo. https://doi.org/10.5281/zenodo.6342352

McKernan, K. J., Helbert, Y., Kane, L. T., Ebling, H., Zhang, L., Liu, B., Eaton, Z., McLaughlin, S., Kingan, S., Baybayan, P., Concepcion, G., Jordan, M., Riva, A., Barbazuk, W., & Harkins, T. (2020). Sequence and annotation of 42 cannabis genomes reveals extensive copy number variation in cannabinoid synthesis and pathogen resistance genes. *BioRxiv*. https://doi.org/10.1101/2020.01.03.894428

Zager, J. J., Lange, I., Srividya, N., Smith, A., & Markus Lange, B. (2019). Gene networks underlying cannabinoid and terpenoid accumulation in cannabis. *Plant Physiology*, *180*(4), 1877–1897. https://doi.org/10.1104/pp.18.01506