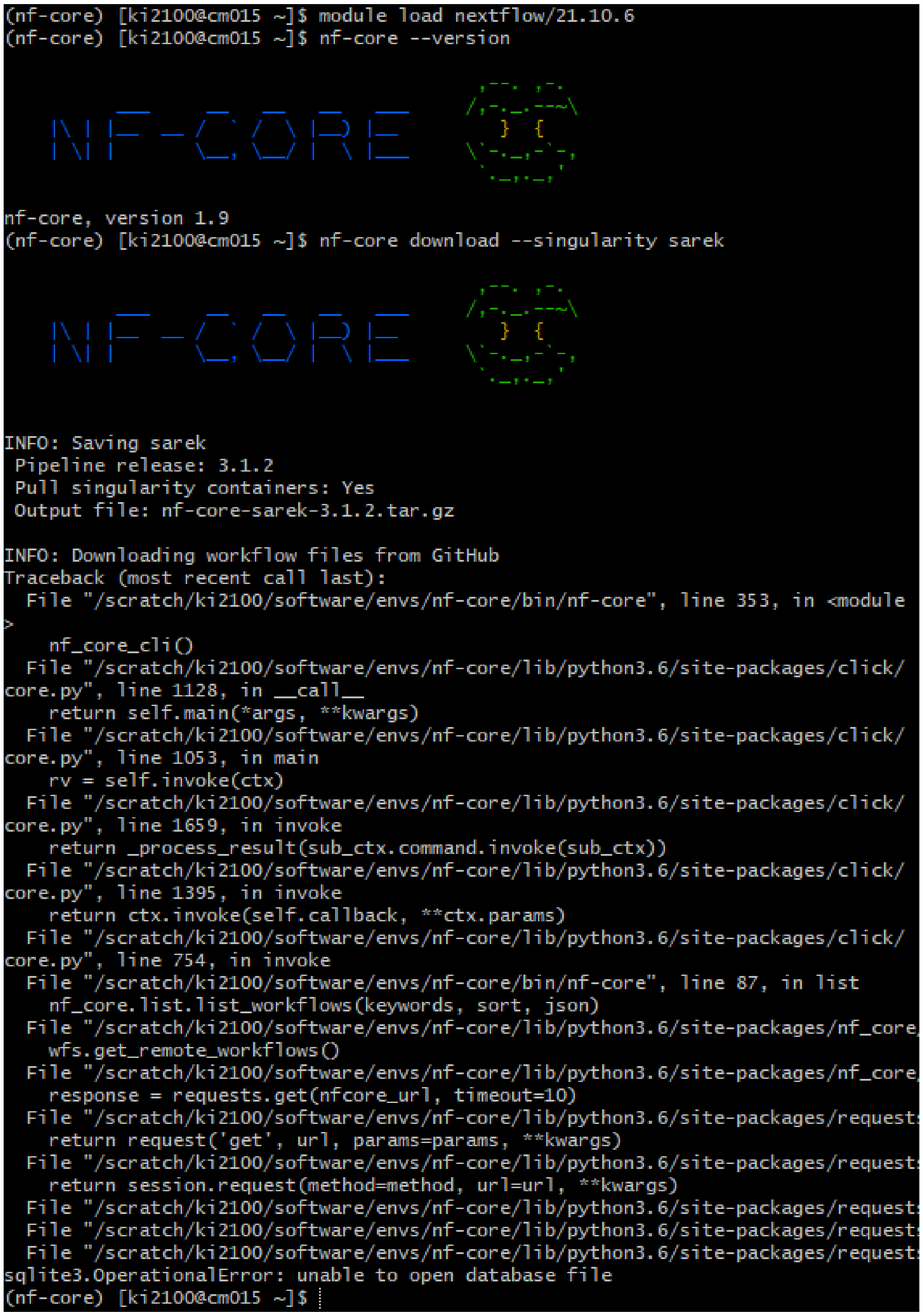
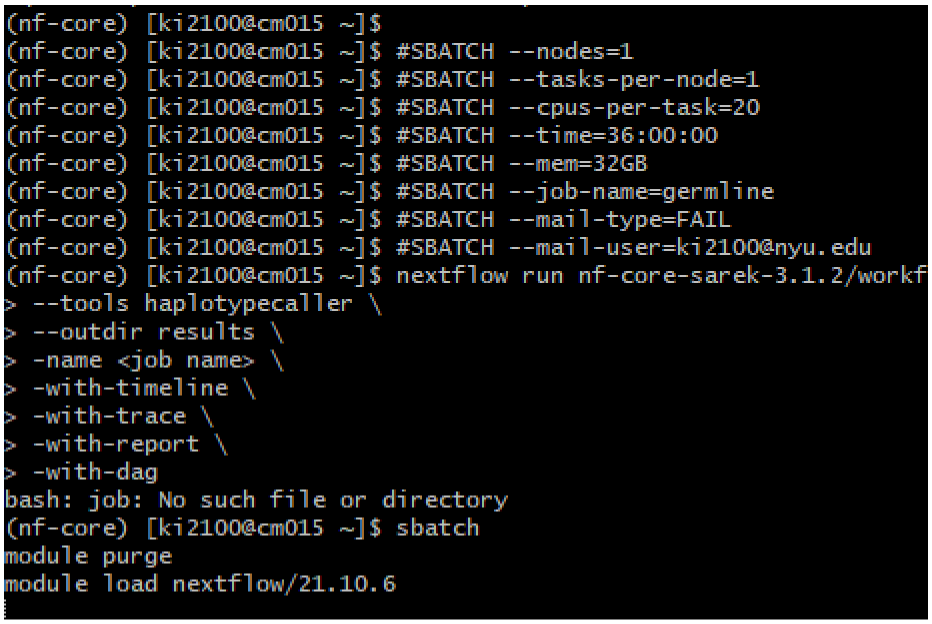
Khan Inan
Assignment week #5

Q2.1. Provide the output of nf-core list for your answer (1 point).



```
(nf-core) [ki2100@cm015 ~]$ module load nextflow/21.10.6
(nf-core) [ki2100@cm015 ~]$ nf-core --version

                                          ,--./,-.
          ___     __   __   __   ___     /,-._.--~\
    |\ | |__  __ /  ` /  \ |__) |__         }  {
    | \| |       \__, \__/ |  \ |___     \`-._,-`-,
                                          `._,._,'

nf-core, version 1.9
(nf-core) [ki2100@cm015 ~]$ nf-core download --singularity sarek

                                          ,--./,-.
          ___     __   __   __   ___     /,-._.--~\
    |\ | |__  __ /  ` /  \ |__) |__         }  {
    | \| |       \__, \__/ |  \ |___     \`-._,-`-,
                                          `._,._,'

INFO: Saving sarek
 Pipeline release: 3.1.2
 Pull singularity containers: Yes
 Output file: nf-core-sarek-3.1.2.tar.gz

INFO: Downloading workflow files from GitHub
Traceback (most recent call last):
  File "/scratch/ki2100/software/envs/nf-core/bin/nf-core", line 353, in <module
>
    nf_core_cli()
  File "/scratch/ki2100/software/envs/nf-core/lib/python3.6/site-packages/click/
core.py", line 1128, in __call__
    return self.main(*args, **kwargs)
  File "/scratch/ki2100/software/envs/nf-core/lib/python3.6/site-packages/click/
core.py", line 1053, in main
    rv = self.invoke(ctx)
  File "/scratch/ki2100/software/envs/nf-core/lib/python3.6/site-packages/click/
core.py", line 1659, in invoke
    return _process_result(sub_ctx.command.invoke(sub_ctx))
  File "/scratch/ki2100/software/envs/nf-core/lib/python3.6/site-packages/click/
core.py", line 1395, in invoke
    return ctx.invoke(self.callback, **ctx.params)
  File "/scratch/ki2100/software/envs/nf-core/lib/python3.6/site-packages/click/
core.py", line 754, in invoke
  File "/scratch/ki2100/software/envs/nf-core/bin/nf-core", line 87, in list
    nf_core.list.list_workflows(keywords, sort, json)
  File "/scratch/ki2100/software/envs/nf-core/lib/python3.6/site-packages/nf_core
    wfs.get_remote_workflows()
  File "/scratch/ki2100/software/envs/nf-core/lib/python3.6/site-packages/nf_core
    response = requests.get(nfcore_url, timeout=10)
  File "/scratch/ki2100/software/envs/nf-core/lib/python3.6/site-packages/request
    return request('get', url, params=params, **kwargs)
  File "/scratch/ki2100/software/envs/nf-core/lib/python3.6/site-packages/request
    return session.request(method=method, url=url, **kwargs)
  File "/scratch/ki2100/software/envs/nf-core/lib/python3.6/site-packages/request
  File "/scratch/ki2100/software/envs/nf-core/lib/python3.6/site-packages/request
  File "/scratch/ki2100/software/envs/nf-core/lib/python3.6/site-packages/request
sqlite3.OperationalError: unable to open database file
(nf-core) [ki2100@cm015 ~]$
```



```
(nf-core) [ki2100@cm015 ~]$
(nf-core) [ki2100@cm015 ~]$ #SBATCH --nodes=1
(nf-core) [ki2100@cm015 ~]$ #SBATCH --tasks-per-node=1
(nf-core) [ki2100@cm015 ~]$ #SBATCH --cpus-per-task=20
(nf-core) [ki2100@cm015 ~]$ #SBATCH --time=36:00:00
(nf-core) [ki2100@cm015 ~]$ #SBATCH --mem=32GB
(nf-core) [ki2100@cm015 ~]$ #SBATCH --job-name=germline
(nf-core) [ki2100@cm015 ~]$ #SBATCH --mail-type=FAIL
(nf-core) [ki2100@cm015 ~]$ #SBATCH --mail-user=ki2100@nyu.edu
(nf-core) [ki2100@cm015 ~]$ nextflow run nf-core-sarek-3.1.2/workf
> --tools haplotypecaller \
> --outdir results \
> -name <job name> \
> -with-timeline \
> -with-trace \
> -with-report \
> -with-dag
bash: job: No such file or directory
(nf-core) [ki2100@cm015 ~]$ sbatch
module purge
module load nextflow/21.10.6
```

Q4.1a Provide the output of your nextflow log command for your script.

The output from the second command provides the directory where the intermediate outputs for each task is written. Note that the pipeline produced a directory called work which, in turn, contains a directories with two letter codes and a subdirectory named using a "hash code" (a unique name based on calculating the hash of a combination of complete file name path, timestamp and file size).

Q4.1b How do we learn more about what analysis produced the outputs in each of these cryptically named directories? Continue reading the Nextflow documentation under the section Execution Log and **execute a command that will return report a table with columns "hash,name,exit, and current status"**. Note that an "exit status" of 0 for each task indicates the task completed without any errors.

We can learn more about the analysis by viewing the properties or details of the directories

For your answer to 4.1b, provide your command line to produce output with "hash,name,exit, and current status" columns and the first 5 lines of output (1 point)

4.1c Next, use tail command to review the last lines of your slurm-.out file for the Sarek pipeline job run.

My job had some errors because I was unable to fully execute due to some disc quota errors

For your answer, determine whether your job completed successfully and without errors. Comment on how each the STDERR and STDOUT of your job (in slurm-.out), the nextflow log output, and your answer to Q4.1b help in establishing your conclusion.

Q4.2. The caching of intermediate results (in the work directory) allows Nextflow pipelines to be restarted at the problematic point should any step fail due to some HPC issue (e.g., node crash)

Read this blog on the Nextflow resuming a pipeline that completed with errors:
https://www.nextflow.io/blog/2019/demystifying-nextflow-resume.html

If your job had not completed successfully, what command line option would you add to resume the job? (1 point)

I would most likely have to use nextflow run followed by nextflow resume

Q5.1. Read mapping rate (i.e., percent mapping rate) to a reference is an indicator of library quality and/or the libraries divergence from the reference genome. Answer the following about read mapping rate:

Q5.1a How many reads were mapped in the file NA12878.recal.cram?

I found there to be nearly 300000 reads mapped

Q5.1b How many reads were unmapped?

0

Q5.1c What is the reported percentage of mapped reads (the “read mapping rate”)?

99%

Q5.2 In Week 4 you learned that the base quality scores (PHRED-scaled error probabilities) that are reported by Illumina and other sequencing platforms are uncalibrated. The GATK base quality score recalibration (BQSR) adjusts these error probabilities in the aligned reads based on a machine learning procedure that uses a known database of true SNPs. Now answer the following (1 point)

Q5.2a Which of the following best describes the uncalibrated (“reported”) base qualities compared to the calibrated/adjusted (“empirical”) base qualities.

The uncalibarated base qualities are more inaccurate compared to the calibrated base qualities

Select the single best answer concerning your observations across the range of base quality scores (6 to 35):

(a) the reported quality scores are always **lower** than the empirical quality scores

(b) the reported quality scores are always **higher** than the empirical quality scores

(c) the reported quality scores are either **equal to or lower than** empirical quality scores

(d) the reported quality scores are either **equal to or higher than** empirical quality scores

Answer choice D seems to be the most accurate out of the other choices since the reported scores seem to be higher

Q5.2b What does your answer indicate about the error rate in base-calling in the raw (reported) read data.

a. the reported base-calling error rates are over-estimated in all quality score classes
b. the reported base-calling error rates are over-estimated in at least some quality score classes
c. the reported base-calling error rates are under-estimated in all quality score classes
d. the reported base-calling error rates are under-estimated in at least some quality score classes

Answer choice B is the most accurate since the system seems to err on the side of caution by over-estimating error rates

Q5.3 Coverage depth captures the average number of shorts reads aligned to each position on each chromosome/contig in the reference genome (or can also be expressed as a single genome-wide value). Using the coverage depth information Mosdepth output section “Average coverage per contig”, do you think individual NA12878 is a male or female? Provide evidence to support your answer (1 point)

I believe that individual na12878 is female based on the coverage depth information

Q5.5. The nextflow log command described above provided some insight into the processes that were executed, but the order of steps is not explicit from that view. Review the report named “execution_timeline_.html” and summarize the order steps. Each step should appear once in your answer, even though some steps are reported multiple times because these steps were broken into multiple smaller steps by Sarek for efficiency of processing (1 point).

From what I understand the file execution_timeline_html is essentially a log for all the scripts/code ran in the current session. The steps that are reported here reflect user input and some steps have been broken down due to efficiency and keeping things under system constraints like memory

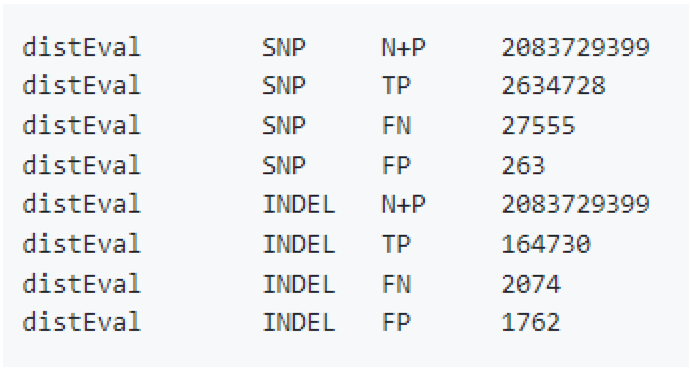
Q5.6. The primary output of interest is the final VCF found here, which has both SNPs and short indels. results/variant_calling/haplotypecaller/NA12878/NA12878.haplotypecaller.filtered.vcf.gz. You may wish to review the file by cd-ing to the haplotypecaller directory and doing:

gunzip -c NA12878.haplotypecaller.filtered.vcf.gz | less # then jump between screens using "d" key until you start seeing variants (lines that dont begin with # or ##)

This call set has had filters applied (i.e., variant records/rows passing or failing filters are both found in this file) in the FILTER column. **note: Analysis of these SNPs would typically first require removing the filtered variants from the callset prior to downstream analysis.**

Sarek runs vcftools software to generate a summary of the filters applied to the variant call set. The output file reports/vcftools/haplotypecaller/NA12878/NA12878.haplotypecaller.filtered.FILTER.summary provides counts of the numbers of variants that passed or failed the filters incorporated into NA12878.haplotypecaller.filtered.vcf.gz.

Q5.6a Report the counts of variants that passed filters, the names of the filters, and the number of variants that failed each filter (1 point).



```
distEval        SNP     N+P     2083729399
distEval        SNP     TP      2634728
distEval        SNP     FN      27555
distEval        SNP     FP      263
distEval        INDEL   N+P     2083729399
distEval        INDEL   TP      164730
distEval        INDEL   FN      2074
distEval        INDEL   FP      1762
```

We can see that the less constraining filters allowed for the highest amount of passed variants while as we go further down the list to the more constraining filters, less and less variants were allowed to pass through