Bioinformatics Bootcamp

Outline

- TSCC Tips
- Variant Calling
- Variant Filtering + Annotation
- Variant Prioritization and Follow-up

TSCC Nodes and Storage

Node	Cores	RAM	Walltime
Login	n/a	n/a	Do not run long jobs here
Hotel	16	64GB	168 hours
Condo	16-28	64-256GB	8 hours
Home (5 nodes)	24-28	128-256GB	unlimited

Location	Capacity	Notes
/home/$USER/	100GB	Slow I/O. Do not write I/O heavy jobs here.
/oasis/tscc/scratch/$USER	~25Tb ($ lfs quota -u $USER /oasis/tscc/scratch –h)	Fast I/O. Slows down when there are lots of files in one directory

Setting up Anaconda (recommended)

- Download Anaconda3
  - `$ wget https://repo.anaconda.com/archive/Anaconda3-2019.07-Linux-x86_64.sh; bash Anaconda3-2019.07-Linux-x86_64.sh`
  - Follow the instructions and agree to append commands to your ~/.bashrc

- Anaconda allows for easy installation of packages and environment control
  - Python2 tools vs python3. For example leafcutter uses python2.7 and R3.3.3
  - `$ conda create –n leafcutter python=2.7 R=3.3.3`
  - `$ conda activate leafcutter`
  - `… follow installation instruction`

- Nowadays almost all tools can be installed with conda. Good rule of thumb is to search “conda install <tool name>” before trying to manually install it

Organization and Submitting Jobs with PBS

- `$ man qsub`
- I recommend this method for organization and submitting jobs. I found that it allows me to quickly refer back to jobs I ran, even years ago.

- For each project directory, I have three folders:
  - logs : log files go here
  - batch : qsub command files and scripts go here
  - src : scripts go here

```
/home/dantakli/my_project/
├── batch
├── logs
└── src
```

- For this example we will submit a job that executes $ echo “hello world ${number}” for numbers 1-10

Qsub example

1. Write a script that writes commands
   - This may sound dumb, but it provides a record of how you generated your commands and allows for quick editing and automation

```
#!/usr/bin/env python
for x in range(1,11): print("echo \"hello world {}\"".format(x))
```

2. Save the output from 1. as a plain-text file
   - `$ python src/make_cmds.py >batch/example_cmd`
3. Make the qsub script with pbsmaker
   - Install with pip:
     - `$ pip install` https://github.com/dantaki/pbsmaker/releases/download/0.0.4/pbsmaker-0.0.4.tar.gz
     - **Documentation**: https://github.com/dantaki/pbsmaker
   - Pbsmaker is a tool that makes qsub scripts. It has a lot of options and flexibility
   - `pbsmaker -i /home/dantakli/my_project/batch/example_cmd  -q home -t 0:30:00 -n example -o /home/dantakli/my_project/logs/ >batch/example.qsub`

- 4. Submit the job `$ qsub batch/example.qsub`

Qsub example cont.

- After running the job, the log files will be written to the log directory (pbsmaker option –o)

- To run jobs in parallel, use job arrays.
  - `pbsmaker -i /home/dantakli/my_project/batch/example_cmd  -q home -t 0:30:00 -n example -o /home/dantakli/my_project/logs/ -T 1-10 –B 10 >batch/example.jobarray.qsub`

  - The –T and –B options indicate to run jobs in parallel
    - -T : run commands from LINE_START – LINE_END (also –T 1-5,8,9 is acceptable format to run jobs 1 through 5, 8, and 9)
    - -B : number of jobs to run in parallel

- You can check on your jobs with $ qstat –u $USER –t
  - Q : Queued
  - R : Running
  - E : Exiting
  - H : Holding

Last Tips: hacking your ~/.bashrc

- You will need Java for GATK
  - `export PATH=/home/dantakli/java/jre1.8.0_73:$PATH`
  - `export PATH=/home/dantakli/java/jre1.8.0_73/bin:$PATH`

- Add your bashrc and account to pbsmaker
  - `alias pbsmaker="pbsmaker -rc /home/$USER/.bashrc -A jogleeson-group ”`

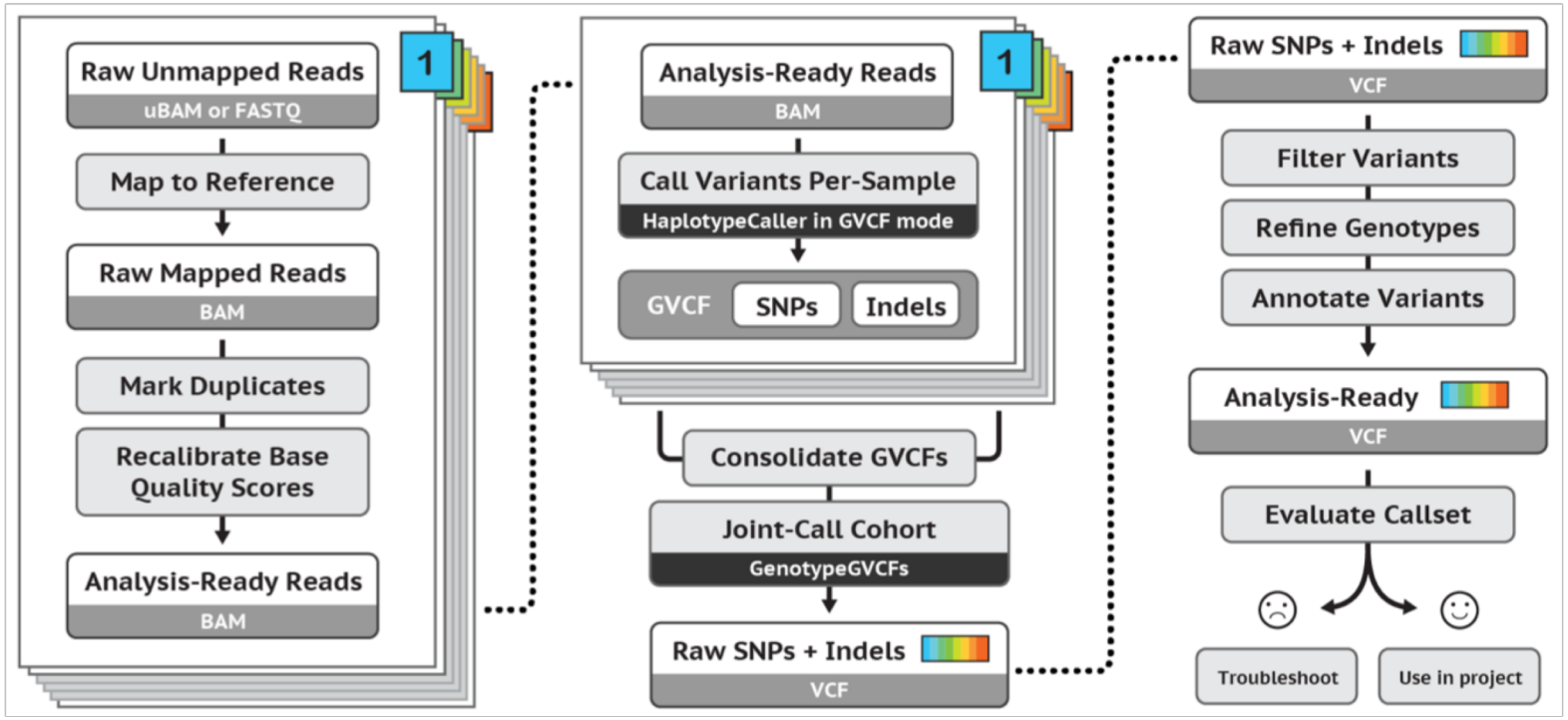
- Check your submitted jobs
  - `alias qs="qstat -u $USER -t”`

- Shortcut for OASIS
  - `export OASIS=/oasis/tscc/scratch/$USER/`
  - Use like this: `$ cd $OASIS`

Variant Calling

- SNP – INDEL with GATK haplotype caller
- https://software.broadinstitute.org/gatk/best-practices/

- You may want to run GATK BQSR (Base Quality Score Recalibration) before variant calling



Variant Filtering and Annotation

- GATK HaplotypeCaller

0. BQSR
1. Sample GVCF
2. Combine GVCFS
3. Joint Genotype GVCFs
4. Recalibrate Variants

```
/home/dantakli/bin/GenomeAnalysisTK-3.8-1/GenomeAnalysisTK.jar
```

Consider learning snakemake for running jobs? Previous members have snakemake pipelines that can automate the following commands.

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

BQSR

```
java –Xmx64G -jar {gatk.jar} \
-T BaseRecalibrator \
-nct {threads} \
-R {ref.fa} \
-I {input.bam} \
-knownSites {dbSNP.vcf} \
-knownSites {Mills.indel.vcf} \
-knownSites {1000G.phase1.indel.vcf} \
-o {output.recal_table}
```

BQSR can be threaded (use at least 8 CPUs)

When threaded it can run under 8 hours.

knownSites found here:
/projects/ps-gleesonlab5/resources/gatk_grch37/

BQSR Print Reads

```
java –Xmx64G -jar {gatk.jar} \
-T PrintReads \
-nct {threads} \
-allowPotentiallyMisencodedQuals \
-R {ref.fa} \
-I {input.bam} \
-BQSR {input.recal_table} \
-o {output.bam}
```

Print reads can take between 1-2 days depending on the size of the input BAM.

Sample GVCF

```
java -Xmx32G -jar {gatk.jar} \
-T HaplotypeCaller \
-R {ref.fa}  \
-I {input.bam} \
--genotyping_mode DISCOVERY \
--doNotRunPhysicalPhasing \
--emitRefConfidence GVCF \
--out {output.gvcf} \
--dbsnp {dbsnp.vcf} \
-log {log.txt} \
--variant_index_type LINEAR \
--variant_index_parameter 128000
```

Haplotype Caller can take a long time to run (2 days). To speed it up you can split the analysis by chromosome with the `-L {chromosome}` option

You will need to supply many CPUs. Probably between 8-16 depending on the requested memory

Resource files:
dbSNP hg19: /projects/ps-gleesonlab5/resources/gatk_grch37/dbsnp_138.b37.vcf
dbSNP hg38: /projects/ps-gleesonlab5/resources/gatk_grch38/dbsnp_146.hg38.vcf.gz

Combine GVCFs

```
java –Xmx64G -jar {gatk.jar} \
-T CombineGVCFs \
-R {ref.fa} \
--variant {input.gvcf_list} \
-o {output.gvcf} \
-log {log} \
```

You will need to supply many CPUs. Probably between 16-24 depending on the requested memory and might need many days to run, depending on the size of your cohort

Joint Genotyping

```
java –Xmx64G -jar {gatk.jar} \
-T GenotypeGVCFs \
-R {ref.fa} \
--variant {input.gvcf} \
--dbsnp {dbsnp.vcf } \
-o {output.vcf} \
-log {log} \
-nt {params.threads} \
--max_alternate_alleles 6
```

Variant Quality Score Recalibration

- VQSR is run into two batches, for SNPs and for INDELs
- Similar to BQSR, there is a recalibration step and a print step
- VQSR is much faster than HaplotypeCaller or BQSR

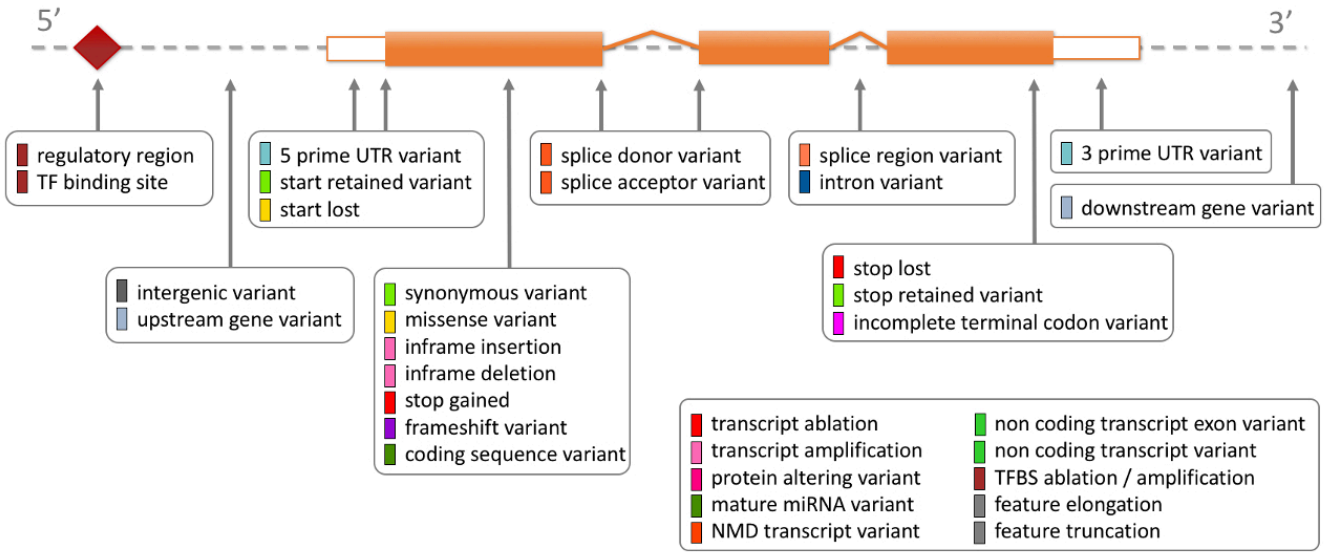
https://github.com/dantaki/GLG/blob/master/pipelines/wes-seq/Snakefile

rule vcf_snp_recalibration:

rule vcf_indel_recalibration:

Variant Annotation with VEP

```
vep --fork 4 -i in.vcf \
--offline –cache \
 –gff  in.gff3.gz \
 --fasta ref.fa \
--sift b --polyphen b  \
--symbol  --uniprot   \
--af_gnomad  \
--show_ref_allele \
--terms SO \
--total_length  --numbers \
--regulatory --variant_class –protein \
-o annotated.txt
```



https://uswest.ensembl.org/info/genome/variation/prediction/predicted_data.html#consequences

Filtering after annotation

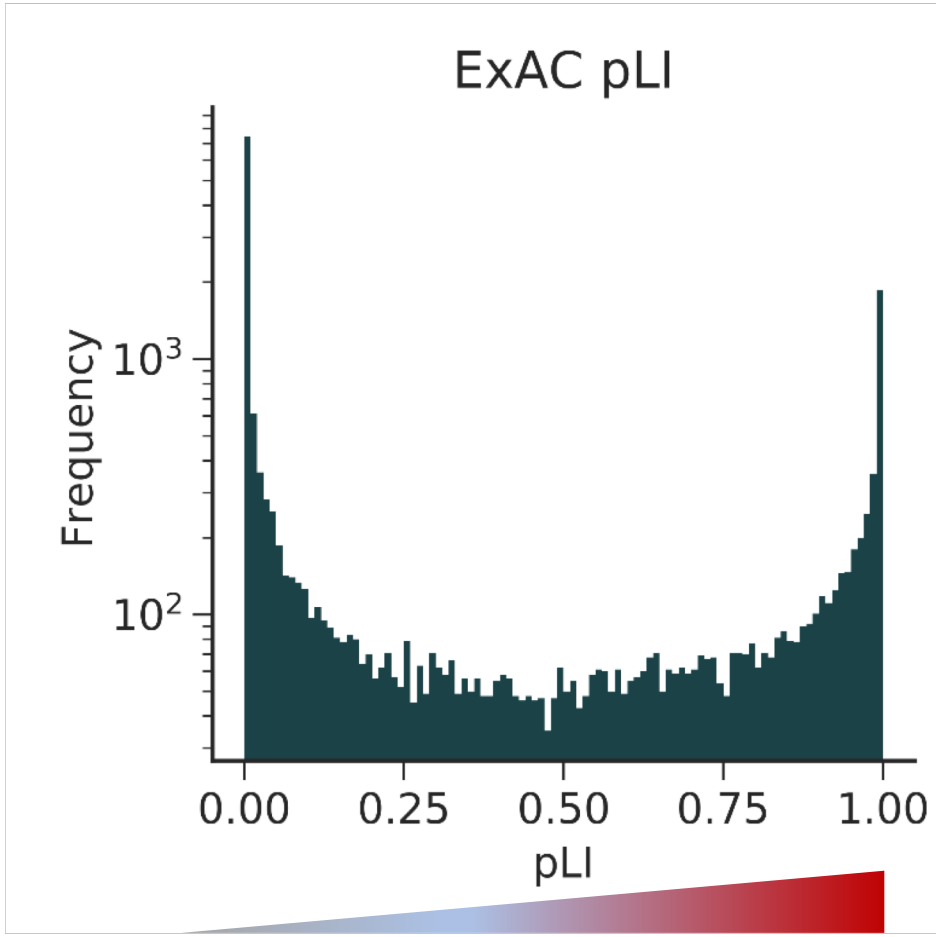
- Omit variants with gnomAD allele frequencies above 1%

- Only consider Loss of Function variants

*	SO term	SO description	SO accession	Display term	IMPACT
	transcript_ablation	A feature ablation whereby the deleted region includes a transcript feature	SO:0001893	Transcript ablation	HIGH
	splice_acceptor_variant	A splice variant that changes the 2 base region at the 3' end of an intron	SO:0001574	Splice acceptor variant	HIGH
	splice_donor_variant	A splice variant that changes the 2 base region at the 5' end of an intron	SO:0001575	Splice donor variant	HIGH
	stop_gained	A sequence variant whereby at least one base of a codon is changed, resulting in a premature stop codon, leading to a shortened transcript	SO:0001587	Stop gained	HIGH
	frameshift_variant	A sequence variant which causes a disruption of the translational reading frame, because the number of nucleotides inserted or deleted is not a multiple of three	SO:0001589	Frameshift variant	HIGH
	stop_lost	A sequence variant where at least one base of the terminator codon (stop) is changed, resulting in an elongated transcript	SO:0001578	Stop lost	HIGH
	start_lost	A codon variant that changes at least one base of the canonical start codon	SO:0002012	Start lost	HIGH
	transcript_amplification	A feature amplification of a region containing a transcript	SO:0001889	Transcript amplification	HIGH

Haploinsufficiency

- I recommend using ExAC pLI scores omitting psychiatric samples
  - gnomAD has not released the pLI scores for non-psychiatric samples
- Retain on 90pc (pLI>=0.99)
- Or gnomAD recommendation
  - pLI >=0.9



Functional Constraint

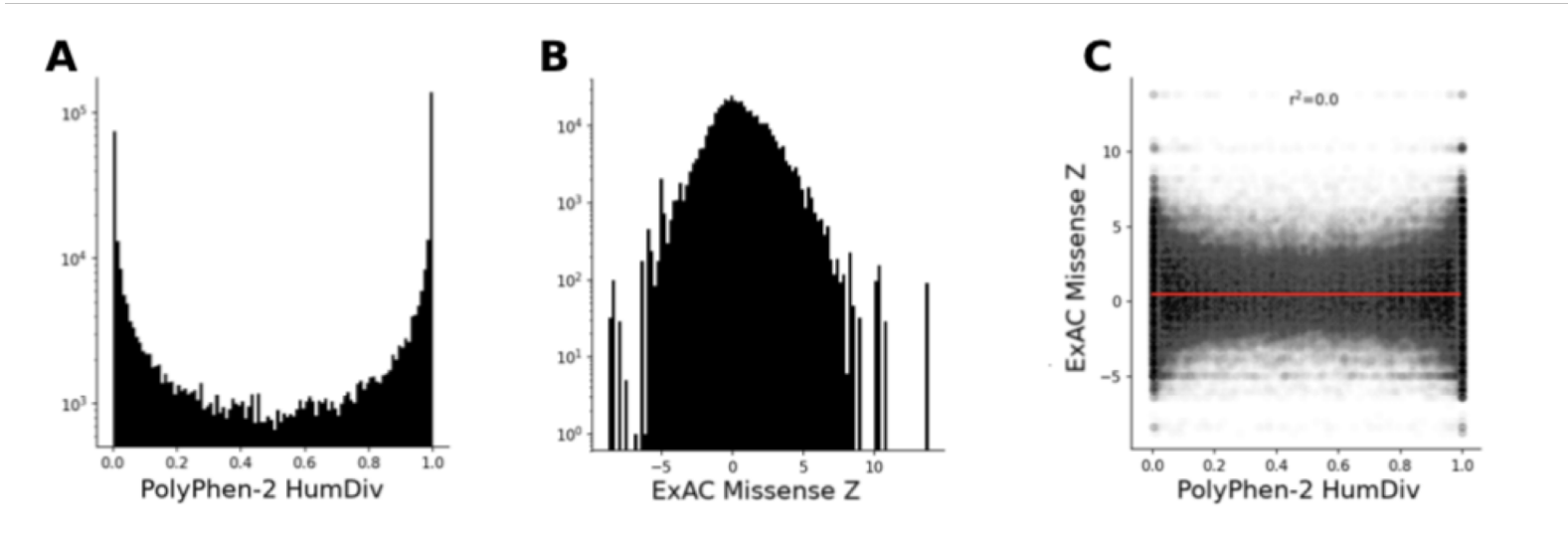
Missense Variants

- Not as clear-cut as LoFs

- PolyPhen-2 scores
  - >=0.957 recommended

- ExAC Missense Z scores
  - Gene based
  - ≥5 sigma (1.33% of genes in ExAC)

- Constrained Coding Region
  - Havrilla 2019 Nat.Gen.
  - Region based
  - >=0.9 recommended



ASD Genes

- SFARI Genes