Variant calling

There are several strategies for variant calling.

Samtools/BCFTools SNP and INDEL calling

Workflows from the htslib

```
#SBATCH -p batch -N 1 -n 4 --mem 16gb
module unload perl
module load samtools
module load bcftools
GENOME=S_enterica_CT18.fasta

# need to make a string which is all the bam files you want to process
# but if we do *.bam it will catch the intermediate bam files that are in the folder
for a in $(cat acc.txt)
do
  m="$a.bam $m"
done

VCF=Salmonella.vcf.gz
VCFFILTER=Salmonella.filtered.vcf.gz
bcftools mpileup -Ou -f $GENOME $m | bcftools call -vmO z -o $VCF
tabix -p vcf $VCF
bcftools stats -F $GENOME -s - $VCF > $VCF.stats
mkdir -p plots
plot-vcfstats -p plots/ $VCF.stats
bcftools filter -O z -o $VCFFILTER -s LOWQUAL -i'%QUAL>10' $VCF
```

Advanced - GATK variant calling

An existing framework that works can be checked out from https://github.com/biodataprog/GEN220_2022_examples see the Variants folder.

Make sure you are running this in ~/bigdata or somewhere with enough space as this will generate large files.

The first script `pipeline_GATK/00_index.sh` will download the genome, index and download the fastq files from NCBI SRA. If you had different datasets you would develop your own data files and script.

you don't need to copy this code - do the git checkout (eg `cd ~/bigdata; git clone https://github.com/biodataprog/GEN220_2022_examples; cd Variants`) and then you can run these steps.

This has a configuration file which defines some variables used by the pipeline.

```
GENOMEFOLDER=genome
```

```
REFGENOME=genome/FungiDB-39_AfumigatusAf293_Genome.fasta
GENOMENAME=Af293
SAMPFILE=samples.csv
FASTQFOLDER=input
FASTQEXT=fastq.gz
UNMAPPED=unmapped
UNMAPPEDASM=unmapped_asm
ASMEXT=fasta
ALNFOLDER=aln
ALNTOOL=bwa
HTCFOLDER=cram
HTCEXT=cram
HTCFORMAT=cram
TEMP=cram
GVCFFOLDER=Variants
VARIANTFOLDER=Variants
TREEDIR=strain_tree
RGCENTER=NCBI
RGPLATFORM=Illumina
GVCF_INTERVAL=1
FINALVCF=vcf
PREFIX=Afum_v1
REFNAME=AF293-REF
SLICEVCF=vcf_slice
SNPEFFOUT=snpEff
snpEffConfig=snpEff.config
SNPEFFGENOME=AfumigatusAf293_FungiDB
GFFGENOME=FungiDB-49_AfumigatusAf293.gff
```

You can customize that as you need for your own data.

The `samples.csv` has a header and is 3 samples for strains from the fungus Aspergillus fumigatus.

```
Strain,Filebase
CEA10,SRR7418934
ISSF_21,SRR4002443
1F1SW_F4,SRR4002444
IFM_60237,DRR022927
```

1. **pipeline_GATK/00_index.sh**

Run this as `sbatch pipeline_GATK/00_index.sh` - you will need to wait for it to finish before doing step 2. This step uses the 4 strains which are already on the cluster. Or if you are on your own computer and have the sratoolkit (fastq-dump) installed it will download that.

```
#!/usr/bin/bash
module load samtools/1.11
module load bwa/0.7.17
if [ -f config.txt ]; then
    source config.txt
fi
mkdir -p $FASTQFOLDER $GENOMEFOLDER
pushd $GENOMEFOLDER
# THIS IS EXAMPLE CODE FOR HOW TO DOWNLOAD DIRECT FROM FUNGIDB
RELEASE=49
SPECIES=AfumigatusAf293
URL=https://fungidb.org/common/downloads/release-${RELEASE}/$SPECIES
PREF=FungiDB-${RELEASE}_${SPECIES}
FASTAFILE=${PREF}_Genome.fasta
DOMAINFILE=${PREF}_InterproDomains.txt
GFF=${PREF}.gff
## THIS IS FUNGIDB DOWNLOAD PART
echo "working off $FASTAFILE - check if these don't match may need to update config/init scr

if [ ! -f $DOMAINFILE ]; then
    curl -O $URL/txt/$DOMAINFILE
fi
if [ ! -f $FASTAFILE ] ; then
    curl -O $URL/fasta/data/$FASTAFILE
fi
if [ ! -f $GFF ]; then
    curl -O $URL/gff/data/$GFF
fi

if [[ ! -f $FASTAFILE.fai || $FASTAFILE -nt $FASTAFILE.fai ]]; then
    samtools faidx $FASTAFILE
fi
if [[ ! -f $FASTAFILE.bwt || $FASTAFILE -nt $FASTAFILE.bwt ]]; then
    bwa index $FASTAFILE
fi

DICT=$(basename $FASTAFILE .fasta)".dict"

if [[ ! -f $DICT || $FASTAFILE -nt $DICT ]]; then
    rm -f $DICT
    samtools dict $FASTAFILE > $DICT
    ln -s $DICT $FASTAFILE.dict
fi

popd
module load sratoolkit
IFS=,
tail -n +2 $SAMPFILE | while read STRAIN SRA
```

```
do
    echo $STRAIN $SRA
    # if On UCR HPCC can use already downloaded files
    if [ ! -f $FASTQFOLDER/${SRA}_1.$FASTQEXT ]; then
        if [ -f /bigdata/gen220/shared/data/Afum/${SRA}_1.$FASTQEXT ]; then
            ln -s /bigdata/gen220/shared/data/Afum/${SRA}_[12].$FASTQEXT $FASTQFOLDER
        else
            fastq-dump -O $FASTQFOLDER --split-e --gzip $SRA
        fi
    fi
done
```

2. **pipeline_GATK/01_index.sh**

Run this as

```
sbatch -a 1-4 pipeline_GATK/01_align.sh
```

You will need to wait for it to finish before doing step 3. This step uses the 4 strains. The code at the bottom is for generating a dataset of unaligned reads for further assembly alone.

```
#!/bin/bash
#SBATCH -N 1 -n 16 --mem 32gb --out logs/bwa.%a.log --time 8:00:00
module load bwa
module load samtools/1.11
module load picard
module load gatk/4
module load java/13

MEM=32g

TMPOUTDIR=tmp

if [ -f config.txt ]; then
  source config.txt
fi
if [ -z $REFGENOME ]; then
  echo "NEED A REFGENOME - set in config.txt and make sure 00_index.sh is run"
  exit
fi

if [ ! -f $REFGENOME.dict ]; then
  echo "NEED a $REFGENOME.dict - make sure 00_index.sh is run"
fi
mkdir -p $TMPOUTDIR $ALNFOLDER

CPU=2
```

```
if [ $SLURM_CPUS_ON_NODE ]; then
  CPU=$SLURM_CPUS_ON_NODE
fi
N=${SLURM_ARRAY_TASK_ID}
if [ -z $N ]; then
  N=$1
fi
if [ -z $N ]; then
  echo "cannot run without a number provided either cmdline or --array in sbatch"
  exit
fi

MAX=$(wc -l $SAMPFILE | awk '{print $1}')
if [ $N -gt $MAX ]; then
  echo "$N is too big, only $MAX lines in $SAMPFILE"
  exit
fi

IFS=,
tail -n +2 $SAMPFILE | sed -n ${N}p | while read STRAIN FILEBASE
do

  # BEGIN THIS PART IS PROBABLY PROJECT SPECIFIC
  # THIS COULD NEED TO BE CHANGED TO R1 R2 or R1_001 and R2_001 etc
  PAIR1=$FASTQFOLDER/${FILEBASE}_1.$FASTQEXT
  PAIR2=$FASTQFOLDER/${FILEBASE}_2.$FASTQEXT
  PREFIX=$STRAIN
  # END THIS PART IS PROBABLY PROJECT SPECIFIC
  echo "STRAIN is $STRAIN $PAIR1 $PAIR2"

  TMPBAMFILE=$TMPOUTDIR/$STRAIN.unsrt.bam
  SRTED=$TMPOUTDIR/$STRAIN.srt.bam
  DDFILE=$TMPOUTDIR/$STRAIN.DD.bam
  FINALFILE=$ALNFOLDER/$STRAIN.$HTCEXT

  READGROUP="@RG\tID:$STRAIN\tSM:$STRAIN\tLB:$PREFIX\tPL:illumina\tCN:$RGCENTER"

  if [ ! -s $FINALFILE ]; then
    if [ ! -s $DDFILE ]; then
      if [ ! -s $SRTED ]; then
        if [ -e $PAIR1 ]; then
          if [ ! -f $TMPBAMFILE ]; then
            # potential switch this to bwa-mem2 for extra speed
            bwa mem -t $CPU -R $READGROUP $REFGENOME $PAIR1 $PAIR2 | samtools view -1 -o $TM
          fi
        else
```

```
          echo "Cannot find $PAIR1, skipping $STRAIN"
          exit
        fi
        samtools fixmate --threads $CPU -O bam $TMPBAMFILE $TEMP/${STRAIN}.fixmate.bam
        samtools sort --threads $CPU -O bam -o $SRTED -T $TEMP $TEMP/${STRAIN}.fixmate.bam
        if [ -f $SRTED ]; then
          rm -f $TEMP/${STRAIN}.fixmate.bam $TMPBAMFILE
        fi
      fi # SRTED file exists or was created by this block

      time java -jar $PICARD MarkDuplicates I=$SRTED O=$DDFILE \
        METRICS_FILE=logs/$STRAIN.dedup.metrics CREATE_INDEX=true VALIDATION_STRINGENCY=SILE
      if [ -f $DDFILE ]; then
        rm -f $SRTED
      fi
    fi # DDFILE is created after this or already exists

    samtools view -O $HTCFORMAT --threads $CPU --reference $REFGENOME -o $FINALFILE $DDFILE
    samtools index $FINALFILE

    if [ -f $FINALFILE ]; then
      rm -f $DDFILE
      rm -f $(echo $DDFILE | sed 's/bam$/bai/')
    fi
  fi #FINALFILE created or already exists


  # The rest of this could be skipped as it is for a project to extrat the UNMAPPED reads a
  FQ=$(basename $FASTQEXT .gz)
  UMAP=$UNMAPPED/${STRAIN}.$FQ
  UMAPSINGLE=$UNMAPPED/${STRAIN}_single.$FQ
  #echo "$UMAP $UMAPSINGLE $FQ"

  if [ ! -f $UMAP ]; then
    module load BBMap
    samtools fastq -f 4 --threads $CPU -N -s $UMAPSINGLE -o $UMAP $FINALFILE
    pigz $UMAPSINGLE
    repair.sh in=$UMAP out=$UMAP.gz
    unlink $UMAP
  fi
done
```

3. This step converts the .cram files (which are BAM files but in a more compressed format) into g.vcf files which are for calling all possile variants. you run it again with array jobs and one job per strain.

```
sbatch -a 1-4 pipeline_GATK/02_call_gvcf.sh
```

```bash
#!/usr/bin/bash
#SBATCH -p intel -N 1 -n 16 --mem 32gb --out logs/make_gvcf.%a.log --time 48:00:00

module load picard
module load java/13
module load gatk/4
module load bcftools

MEM=32g
SAMPFILE=samples.csv

if [ -f config.txt ]; then
    source config.txt
fi

DICT=$(echo $REFGENOME | sed 's/fasta$/dict/')

if [ ! -f $DICT ]; then
    picard CreateSequenceDictionary R=$GENOMEIDX O=$DICT
fi
mkdir -p $VARIANTFOLDER
CPU=1
if [ $SLURM_CPUS_ON_NODE ]; then
 CPU=$SLURM_CPUS_ON_NODE
fi
N=${SLURM_ARRAY_TASK_ID}

if [ ! $N ]; then
 N=$1
fi

if [ ! $N ]; then
 echo "need to provide a number by --array slurm or on the cmdline"
 exit
fi

hostname
date
IFS=,
tail -n +2 $SAMPFILE | sed -n ${N}p | while read STRAIN SAMPID
do
  # BEGIN THIS PART IS PROJECT SPECIFIC LIKELY
  # END THIS PART IS PROJECT SPECIFIC LIKELY
  echo "STRAIN is $STRAIN"
  GVCF=$VARIANTFOLDER/$STRAIN.g.vcf
  ALNFILE=$ALNFOLDER/$STRAIN.$HTCEXT
  if [ -s $GVCF.gz ]; then
    echo "Skipping $STRAIN - Already called $STRAIN.g.vcf.gz"
    exit
  fi
  if [[ ! -f $GVCF || $ALNFILE -nt $GVCF ]]; then
      time gatk --java-options -Xmx${MEM} HaplotypeCaller \
      --emit-ref-confidence GVCF --sample-ploidy 1 \
      --input $ALNFILE --reference $REFGENOME \
      --output $GVCF --native-pair-hmm-threads $CPU \
         -G StandardAnnotation -G AS_StandardAnnotation -G StandardHCAnnotation
 fi
 bgzip --threads $CPU -f $GVCF
 tabix $GVCF.gz
done
date
```

4. This step runs GVCF -> final VCF but one chromosome at time. The number of chromosomes are the number of sequencing the genome FASTA file or you can count with `wc -l genome/*.fai`

If your genome is fragmented you will want to adjust the parameter in `config.txt` file so that `GVCF_INTERVAL=1` is more like 5 or 10 and then adjust your job number by that factor. Eg if you have 1000 contigs and `GVCF_INTERVAL=5` then you would want to run array jobs with 1000/5 = 200 instead of 1000 jobs.

```
sbatch -a 1-9 pipeline_GATK/03_jointGVCF_call_slice.sh
```

Here is the Code

```
#!/usr/bin/bash
#SBATCH --mem 24G --nodes 1 --ntasks 2 -J slice.GVCFGeno --out logs/GVCFGenoGATK4.slice_%a.
#--time 48:00:00
hostname
MEM=24g
module unload java
module load picard
module load gatk/4
module load java/13
module load bcftools
module load parallel

source config.txt

declare -x TEMPDIR=$TEMP/$USER/$$

cleanup() {
    #echo "rm temp is: $TEMPDIR"
    rm -rf $TEMPDIR
    rmdir $TEMPDIR
}

# Set trap to ensure cleanupis stopped
trap "cleanup; rm -rf $TEMPDIR; exit" SIGHUP SIGINT SIGTERM EXIT

GVCF_INTERVAL=1
N=${SLURM_ARRAY_TASK_ID}

if [ -z $N ]; then
    N=$1
    if [ -z $N ]; then
        echo "Need an array id or cmdline val for the job"
        exit
    fi
fi
if [ -f config.txt ]; then
    source config.txt
fi
if [ -z $SLICEVCF ]; then
    SLICEVCF=vcf_slice
fi
mkdir -p $SLICEVCF
STEM=$SLICEVCF/$PREFIX.$N
GENOVCFOUT=$STEM.all.vcf
FILTERSNP=$STEM.SNP.filter.vcf
FILTERINDEL=$STEM.INDEL.filter.vcf
SELECTSNP=$STEM.SNP.selected.vcf
SELECTINDEL=$STEM.INDEL.selected.vcf

if [ ! -f $REFGENOME ]; then
    module load samtools/1.9
    samtools faidx $REFGENOME
fi
NSTART=$(perl -e "printf('%d',1 + $GVCF_INTERVAL * ($N - 1))")
NEND=$(perl -e "printf('%d',$GVCF_INTERVAL * $N)")
MAX=$(wc -l $REFGENOME.fai | awk '{print $1}')
if [ "$NSTART" -gt "$MAX" ]; then
    echo "NSTART ($NSTART) > $MAX"
    exit
fi
if [ "$NEND" -gt "$MAX" ]; then
    NEND=$MAX
fi
echo "$NSTART -> $NEND"
```

```
CPU=$SLURM_CPUS_ON_NODE
if [ ! $CPU ]; then
    CPU=2
fi
if [[ $(ls $GVCFFOLDER | grep -c -P "\.g.vcf$") -gt "0" ]]; then
    parallel -j $CPU bgzip {} ::: $GVCFFOLDER/*.g.vcf
    parallel -j $CPU tabix -f {} ::: $GVCFFOLDER/*.g.vcf.gz
fi

FILES=$(ls $GVCFFOLDER/*.g.vcf.gz | sort | perl -p -e 's/(\S+)\n/-V $1 /')
INTERVALS=$(cut -f1 $REFGENOME.fai  | sed -n "${NSTART},${NEND}p" | perl -p -e 's/(\S+)\n/--
mkdir -p $TEMPDIR
if [ ! -f $GENOVCFOUT.gz ]; then
    if [ ! -f $GENOVCFOUT ]; then
    DB=$TEMPDIR/${GVCFFOLDER}_slice_$N
    rm -rf $DB
    gatk  --java-options "-Xmx$MEM -Xms$MEM" GenomicsDBImport --consolidate --merge-input-i
    #--reader-threads $CPU
    #gatk  --java-options "-Xmx$MEM -Xms$MEM" GenomicsDBImport --genomicsdb-workspace-path
    time gatk GenotypeGVCFs --reference $REFGENOME --output $GENOVCFOUT -V gendb://$DB --tmp
    ls -l $TEMPDIR
    rm -rf $DB
    fi
    if [ -f $GENOVCFOUT ]; then
        bgzip $GENOVCFOUT
        tabix $GENOVCFOUT.gz
    fi
fi
TYPE=SNP
echo "VCF = $STEM.$TYPE.vcf.gz"
if [[ ! -f $STEM.$TYPE.vcf.gz ]]; then
    gatk SelectVariants \
    -R $REFGENOME \
    --variant $GENOVCFOUT.gz \
    -O $STEM.$TYPE.vcf \
    --restrict-alleles-to BIALLELIC \
    --select-type-to-include $TYPE --create-output-variant-index false

    bgzip $STEM.$TYPE.vcf
    tabix $STEM.$TYPE.vcf.gz
fi

if [[ ! -f $FILTERSNP.gz || $STEM.$TYPE.vcf.gz -nt $FILTERSNP.gz ]]; then
    gatk VariantFiltration --output $FILTERSNP \
    --variant $STEM.$TYPE.vcf.gz -R $REFGENOME \
    --cluster-window-size 10  \
    --filter-expression "QD < 2.0" --filter-name QualByDepth \
    --filter-expression "MQ < 40.0" --filter-name MapQual \
    --filter-expression "QUAL < 100" --filter-name QScore \
    --filter-expression "SOR > 4.0" --filter-name StrandOddsRatio \
    --filter-expression "FS > 60.0" --filter-name FisherStrandBias \
    --missing-values-evaluate-as-failing --create-output-variant-index false

#   --filter-expression "MQRankSum < -12.5" --filter-name MapQualityRankSum \
#   --filter-expression "ReadPosRankSum < -8.0" --filter-name ReadPosRank \

    bgzip $FILTERSNP
    tabix $FILTERSNP.gz
fi

if [[ ! -f $SELECTSNP.gz || $FILTERSNP.gz -nt $SELECTSNP.gz ]]; then
    gatk SelectVariants -R $REFGENOME \
    --variant $FILTERSNP.gz \
    --output $SELECTSNP \
    --exclude-filtered --create-output-variant-index false
    bgzip $SELECTSNP
    tabix $SELECTSNP.gz
fi

TYPE=INDEL
if [ ! -f $STEM.$TYPE.vcf.gz ]; then
    gatk SelectVariants \
        -R $REFGENOME \
        --variant $GENOVCFOUT.gz \
        -O $STEM.$TYPE.vcf  --select-type-to-include MIXED --select-type-to-include MNP \
        --select-type-to-include $TYPE --create-output-variant-index false
    bgzip $STEM.$TYPE.vcf
    tabix $STEM.$TYPE.vcf.gz
fi

if [[ ! -f $FILTERINDEL.gz || $STEM.$TYPE.vcf.gz -nt $FILTERINDEL.gz ]]; then
    gatk VariantFiltration --output $FILTERINDEL \
    --variant $STEM.$TYPE.vcf.gz -R $REFGENOME \
    --cluster-window-size 10  -filter "QD < 2.0" --filter-name QualByDepth \
    -filter "SOR > 10.0" --filter-name StrandOddsRatio \
    -filter "FS > 200.0" --filter-name FisherStrandBias \
    -filter "InbreedingCoeff < -0.8" --filter-name InbreedCoef \
    --create-output-variant-index false

#   -filter "ReadPosRankSum < -20.0" --filter-name ReadPosRank \
#   -filter "MQRankSum < -12.5" --filter-name MapQualityRankSum \
    bgzip $FILTERINDEL
    tabix $FILTERINDEL.gz
fi

if [[ ! -f $SELECTINDEL.gz || $FILTERINDEL.gz -nt $SELETINDEL.gz ]]; then
    gatk SelectVariants -R $REFGENOME \
    --variant $FILTERINDEL.gz \
    --output $SELECTINDEL \
    --exclude-filtered --create-output-variant-index false
    bgzip $SELECTINDEL
    tabix $SELECTINDEL.gz
fi

if [ -d $TEMPDIR ]; then
    rmdir $TEMPDIR
fi
```

5. Finally to combine all the slices we use this script which is very fast and combines the slices.

```
sbatch pipeline_GATK/04_combine_vcf.sh
```

6. To get predictions about what the impact of SNPs are, which genes they fall in, and whether they are synonymous or non-synonymous changes.

```
sbatch pipeline_GATK/08_snpEff.sh
```

```
#!/usr/bin/bash
#SBATCH --mem=64G -p batch --nodes 1 --ntasks 2 --out logs/snpEff.log
module unload miniconda2
module load miniconda3
module load snpEff
module load bcftools/1.11
module load tabix
# THIS IS AN EXAMPLE OF HOW TO MAKE SNPEFF - it is for A.fumigatus
SNPEFFGENOME=AfumigatusAf293_FungiDB_39
GFFGENOME=$SNPEFFGENOME.gff

MEM=64g

# this module defines SNPEFFJAR and SNPEFFDIR
if [ -f config.txt ]; then
    source config.txt
fi
GFFGENOMEFILE=$GENOMEFOLDER/$GFFGENOME
FASTAGENOMEFILE=$GENOMEFOLDER/$GENOMEFASTA
if [ -z $SNPEFFJAR ]; then
```

```
 echo "need to defined \$SNPEFFJAR in module or config.txt"
 exit
fi
if [ -z $SNPEFFDIR ]; then
 echo "need to defined \$SNPEFFDIR in module or config.txt"
 exit
fi
# could make this a confi

if [ -z $FINALVCF ]; then
    echo "need a FINALVCF variable in config.txt"
    exit
fi

mkdir -p $SNPEFFOUT
## NOTE YOU WILL NEED TO FIX THIS FOR YOUR CUSTOM GENOME
if [ ! -e $SNPEFFOUT/$snpEffConfig ]; then
    rsync -a $SNPEFFDIR/snpEff.config $SNPEFFOUT/$snpEffConfig
    echo "# AfumAf293.fungidb " >> $SNPEFFOUT/$snpEffConfig
    # CHANGE Aspergillus fumigatus Af293 FungiDB to your genome name and source - though th
    echo "$SNPEFFGENOME.genome : Aspergillus fumigatus Af293 FungiDB" >> $SNPEFFOUT/$snpEff
    chroms=$(grep '##sequence-region' $GFFGENOMEFILE | awk '{print $2}' | perl -p -e 's/\n/
    echo -e "\t$SNPEFFGENOME.chromosomes: $chroms" >> $SNPEFFOUT/$snpEffConfig
    # THIS WOULD NEED SPEIFIC FIX BY USER - IN A.fumigatus the MT contig is called mito_A_f
    echo -e "\t$SNPEFFGENOME.mito_A_fumigatus_Af293.codonTable : Mold_Mitochondrial" >> $SN
    mkdir -p $SNPEFFOUT/data/$SNPEFFGENOME
    gzip -c $GFFGENOMEFILE > $SNPEFFOUT/data/$SNPEFFGENOME/genes.gff.gz
    rsync -aL $REFGENOME $SNPEFFOUT/data/$SNPEFFGENOME/sequences.fa

    java -Xmx$MEM -jar $SNPEFFJAR build -datadir `pwd`/$SNPEFFOUT/data -c $SNPEFFOUT/$snpEf
fi
pushd $SNPEFFOUT
COMBVCF="../$FINALVCF/$PREFIX.SNP.combined_selected.vcf.gz ../$FINALVCF/$PREFIX.INDEL.combi
for n in $COMBVCF
do
 echo $n
 st=$(echo $n | perl -p -e 's/\.gz//')
 if [ ! -f $n ]; then
     bgzip $st
 fi
 if [ ! -f $n.tbi ]; then
     tabix $n
 fi
done
INVCF=$PREFIX.allvariants_combined_selected.vcf
OUTVCF=$PREFIX.snpEff.vcf
```

```
OUTTAB=$PREFIX.snpEff.tab
OUTMATRIX=$PREFIX.snpEff.matrix.tsv
DOMAINVAR=$PREFIX.snpEff.domain_variant.tsv
bcftools concat -a -d both -o $INVCF -O v $COMBVCF
java -Xmx$MEM -jar $SNPEFFJAR eff -dataDir `pwd`/data -v $SNPEFFGENOME $INVCF > $OUTVCF

bcftools query -H -f '%CHROM\t%POS\t%REF\t%ALT{0}[\t%TGT]\t%INFO/ANN\n' $OUTVCF > $OUTTAB

# this requires python3 and vcf script
# this assumes the interpro domains were downloaded from FungiDB and their format - you wil
../scripts/map_snpEff2domains.py --vcf $OUTVCF --domains ../genome/${SNPEFFGENOME}_Interpro

# this requires Python and the vcf library to be installed - should be in the miniconda3 en
../scripts/snpEff_2_tab.py $OUTVCF > $OUTMATRIX
```