Calling variants in diploid or multiploid genomes

Diploid genomes

The initial steps in calling variants for diploid or multi-ploid organisms with NGS data are the same as what we've already seen:

1. Map the raw data to a suitable reference
2. Look for SNPs and small-scale indels

Expectations of output are quite different however, which can add statistical power to uncovering variation in populations or organisms with more than two expected variants at the same location.

Example: The CEU Trio from the 1000 Genomes Project

Many example datasets are available from the 1000 genomes project specifically for method evaluation and training. We'll explore a trio (mom, dad, child). Their accession numbers are NA12892, NA12891, and NA12878 respectively. To make the exercise run more quickly, we'll focus on data only from chromosome 20.

All the data we'll use is located here:

Diploid genome (human) example files
`$BI/ngs_course/human_variation`

This directory contains raw data (the .fastq files), mapped data (the .bam files) and variant calls (the .vcf files). It also contains the subdirectory `ref` with special references.

The 1000 Genomes project is really oriented to producing .vcf files; the file "ceu20.vcf" contains all the latest genotypes from this trio based on abundant data from the project.

.bam files containing a subset of mapped human whole exome data are also available on these three; those are the three files "NA*.bam".

We've pre-run samtools and GATK on each sample individually - those are the *GATK.vcf and *samtools.vcf files.

We've also pre-run samtools and GATK on the trio, resulting in GATK.all.vcf and samtools.all.vcf.

We'll now show the steps for:

1. Mapping
2. Single-sample variant calling with samtools
3. Multiple-sample variant calling with samtools

We'll return to this example data shortly to demonstrate a much more involved tool, GATK, to do the same steps.

Mapping Exercise

Let's use Anna's shell script `align_bwa.sh` to align the fastq files to the hg19 version of the genome, and the python program `launcher_creator.py` to create the job submission script.

Don't forget that for this to work, you need to have appended `$BI/bin` to your path.

```
BI="/corral-repl/utexas/BioITeam"
PATH=$PATH:$BI/bin
export PATH
```

Move into your scratch directory and then try to figure out how to create and `qsub` the `align_bwa.sh` command to align the data file `$BI/ngs_course/human_variation/allseqs_R1.fastq` against the `hg19` reference. Call the output "test".

Make job submission script for mapping & variant calling

```
echo "align_bwa.sh $BI/ngs_course/human_variation/allseqs_R1.fastq test hg19 1 > aln.test.log 2>&1" >
commands
launcher_creator.py -l map_call.sge -n map_call -t 01:00:00 -j commands
module load bwa
module load samtools
qsub map_call.sge
```

Caution: If you are using this example outside an SSC course, you must use the -a option to specify a valid allocation (e.g. BME_2012)

```
login1$ echo "align_bwa.sh $BI/ngs_course/human_variation/allseqs_R1.fastq test hg19 1" > commands
launcher_creator.py -l map_call.sge -n map_call -t 01:00:00 -j commands
Job file has 1 lines.
Using 12 cores.
Launcher successfully created. Type "qsub map_call.sge" to queue your job.
login1$ qsub map_call.sge
------------------------------------------------------------------
-- Welcome to the Lonestar4 Westmere/QDR IB Linux Cluster --
------------------------------------------------------------------
--> Checking that you specified -V...
--> Checking that you specified a time limit...
--> Checking that you specified a queue...
--> Setting project...
--> Checking that you specified a parallel environment...
--> Checking that you specified a valid parallel environment name...
--> Checking that the number of PEs requested is valid...
--> Ensuring absence of dubious h_vmem,h_data,s_vmem,s_data limits...
--> Requesting valid memory configuration (23.4G)...
--> Verifying HOME file-system availability...
--> Verifying WORK file-system availability...
--> Verifying SCRATCH file-system availability...
--> Checking ssh setup...
--> Checking that you didn't request more cores than the maximum...
--> Checking that you don't already have the maximum number of jobs...
--> Checking that you don't already have the maximum number of jobs in queue development...
--> Checking that your time limit isn't over the maximum...
--> Checking available allocation...
--> Submitting job...

Your job 586249 ("map_call") has been submitted
```

Note that the input is paired-end data.

Your directory should have content like this when done (from `ls -lt`):

Mapping output

```
-rw-r--r-- 1 sphsmith G-801020      5732 May 20 23:01 map_call.o586338
-rw------- 1 sphsmith G-801020       392 May 20 23:01 test.flagstat.txt
-rw------- 1 sphsmith G-801020   2175952 May 20 23:01 test.sorted.bam.bai
-rw------- 1 sphsmith G-801020 334782188 May 20 23:01 test.sorted.bam
-rw-r--r-- 1 sphsmith G-801020     13135 May 20 23:00 map_call.e586338
-rw------- 1 sphsmith G-801020 411244396 May 20 23:00 test.bam
-rw------- 1 sphsmith G-801020  45695040 May 20 22:49 test.read2.sai
-rw------- 1 sphsmith G-801020  45372400 May 20 22:39 test.read1.sai
-rw-r--r-- 1 sphsmith G-801020         0 May 20 22:26 map_call.pe586338
```

and `samtools flagstat test.sorted.bam` should yield:

samtools flagstat results

```
Running flagstat...
4546280 + 0 in total (QC-passed reads + QC-failed reads)
0 + 0 duplicates
3992274 + 0 mapped (87.81%:nan%)
4546280 + 0 paired in sequencing
2273140 + 0 read1
2273140 + 0 read2
40290 + 0 properly paired (0.89%:nan%)
3636946 + 0 with itself and mate mapped
355328 + 0 singletons (7.82%:nan%)
44128 + 0 with mate mapped to a different chr
15634 + 0 with mate mapped to a different chr (mapQ>=5)
```

Single-sample variant calling with samtools

We would normally use the BAM file from the previous mapping step to call variants in this raw data. However, for the purposes of this course we will use the actual BAM file provided by the 1000 Genomes Project (from which the `.fastq` file above was derived, leading to some oddities in it). `$BI/ngs_course/human_variation/NA12878.chrom20.ILLUMINA.bwa.CEU.exome.20111114.bam`

With samtools, this is a two-step process:

1. `samtools mpileup` command transposes the mapped data in a sorted BAM file fully to genome-centric coordinates. It starts at the first base on the first chromosome for which there is coverage and prints out one line per base. Each line has information on every base observed in the raw data at that base position along with a lot of auxiliary information depending on which flags are set. It calculates the Bayseian prior probability given by the data, but does not estimate a real genotype.
2. `bcftools` with a few options added uses the prior probability distribution and the data to calculate a genotype for the variants detected.

Here are the commands, piped together:

Calling variants using samtools and bcftools

```
samtools mpileup -uf $BI/ref_genome/fasta/ucsc/ucsc.hg19.fasta \
  $BI/ngs_course/human_variation/NA12878.chrom20.ILLUMINA.bwa.CEU.exome.20111114.bam \
  | bcftools view -vcg - > test.raw.bcf
```

Note that the reference chosen for mpileup must be **exactly** the same as the reference used to create the bam file. The 1000 genomes project has created it's own reference and so the command listed above won't work - we have to use the 1000 genomes reference which is `$BI/ngs_course/human_variation/ref/hs37d5.fa`. We could have chosen another mapper if we'd wanted to though.

Exercise:

Write a modified version of the command above to use the proper reference into a `commands` file, create the job submission script with `launcher_creator.py` and submit the job.

As we did for mapping, we need to submit these to the Lonestar queue:

Submission of variant calling commands

```
echo "samtools mpileup -uf $BI/ngs_course/human_variation/ref/hs37d5.fa \
   $BI/ngs_course/human_variation/NA12878.chrom20.ILLUMINA.bwa.CEU.exome.20111114.bam \
   | bcftools view -vcg - > test.raw.vcf" > commands
launcher_creator.py -l call_vars.sge -n call_vars -t 01:00:00 -j commands
qsub call_vars.sge
```

```
login1$ echo "samtools mpileup -uf /corral-repl/utexas/BioITeam/ngs_course/human_variation/ref/hs37d5.fa \
>   /corral-repl/utexas/BioITeam/ngs_course/human_variation/NA12878.chrom20.ILLUMINA.bwa.CEU.exome.20111114.
bam \
>   | bcftools view -vcg - > test.raw.vcf" > commands
login1$ launcher_creator.py -l call_vars.sge -n call_vars -t 01:00:00 -j commands
Job file has 1 lines.
Using 12 cores.
Launcher successfully created. Type "qsub call_vars.sge" to queue your job.
login1$ qsub call_vars.sge
------------------------------------------------------------------
-- Welcome to the Lonestar4 Westmere/QDR IB Linux Cluster --
------------------------------------------------------------------
--> Checking that you specified -V...
--> Checking that you specified a time limit...
--> Checking that you specified a queue...
--> Setting project...
--> Checking that you specified a parallel environment...
--> Checking that you specified a valid parallel environment name...
--> Checking that the number of PEs requested is valid...
--> Ensuring absence of dubious h_vmem,h_data,s_vmem,s_data limits...
--> Requesting valid memory configuration (23.4G)...
--> Verifying HOME file-system availability...
--> Verifying WORK file-system availability...
--> Verifying SCRATCH file-system availability...
--> Checking ssh setup...
--> Checking that you didn't request more cores than the maximum...
--> Checking that you don't already have the maximum number of jobs...
--> Checking that you don't already have the maximum number of jobs in queue development...
--> Checking that your time limit isn't over the maximum...
--> Checking available allocation...
--> Submitting job...

Your job 586369 ("call_vars") has been submitted
```

After your job completes

If you don't want to wait, CHANGE TO A NEW DIRECTORY and do this:
`ln -s $BI/ngs_course/human_variation/NA12878.chrom20.samtools.vcf test.raw.vcf`
and continue with the rest of these exercises. Remember to return to your job directory if you want to see your actual data.

You can examine some of the variant calls with:

Scanning a vcf file

```
more test.raw.vcf
```

Note the extensive header information, followed by an ordered list of variants.

A `bcftools` auxiliary perl script called `vcfutils.pl` can provide some useful stats. It's safe to run this on the head node since it's quick. This is not part of a standard TACC module but it's in our common $BI/bin directory.

Running vcfutils.pl

```
vcfutils.pl qstats test.raw.vcf
```

```
login1$ vcfutils.pl qstats test.raw.vcf
QUAL #non-indel #SNPs #transitions #joint ts/tv #joint/#ref #joint/#non-indel
186        1011        1011        757        0        2.9803        0.0000        0.0000        2.9803
142        2036        2036        1441        0        2.4218        0.0000        0.0000        2.0059
122        3091        3091        2156        0        2.3059        0.0000        0.0000        2.1029
109        4177        4177        2925        0        2.3363        0.0000        0.0000        2.4259
99.5        5237        5237        3647        0        2.2937        0.0000        0.0000        2.1361
92        6273        6273        4354        0        2.2689        0.0000        0.0000        2.1489
85.5        7328        7328        5105        0        2.2964        0.0000        0.0000        2.4704
79        8352        8352        5811        0        2.2869        0.0000        0.0000        2.2201
74        9369        9369        6497        0        2.2622        0.0000        0.0000        2.0725
69        10553        10553        7311        0        2.2551        0.0000        0.0000        2.2000
65        11782        11782        8176        0        2.2673        0.0000        0.0000        2.3764
61        13059        13059        9090        0        2.2902        0.0000        0.0000        2.5179
57        14280        14280        9945        0        2.2941        0.0000        0.0000        2.3361
53        15301        15301        10656        0        2.2941        0.0000        0.0000        2.2935
48.8        16323        16323        11347        0        2.2803        0.0000        0.0000        2.0876
45        17460        17460        12122        0        2.2709        0.0000        0.0000        2.1409
41.8        18639        18639        12899        0        2.2472        0.0000        0.0000        1.9328
39.8        19660        19660        13572        0        2.2293        0.0000        0.0000        1.9339
37.8        21013        21013        14496        0        2.2243        0.0000        0.0000        2.1538
35.8        22309        22309        15380        0        2.2197        0.0000        0.0000        2.1456
33.8        23568        23568        16251        0        2.2210        0.0000        0.0000        2.2448
31.8        24846        24846        17101        0        2.2080        0.0000        0.0000        1.9860
29.8        26171        26171        17975        0        2.1931        0.0000        0.0000        1.9379
28        27308        27308        18688        0        2.1680        0.0000        0.0000        1.6816
26        28654        28654        19551        0        2.1478        0.0000        0.0000        1.7867
24        29947        29947        20371        0        2.1273        0.0000        0.0000        1.7336
22        31050        31050        21065        0        2.1097        0.0000        0.0000        1.6968
20        32081        32081        21719        0        2.0960        0.0000        0.0000        1.7347
17.1        33251        33251        22446        0        2.0774        0.0000        0.0000        1.6411
13.2        34447        34447        23238        0        2.0732        0.0000        0.0000        1.9604
10.4        35751        35751        24067        0        2.0598        0.0000        0.0000        1.7453
9.53        36772        36772        24724        0        2.0521        0.0000        0.0000        1.8049
8.65        38023        38023        25539        0        2.0457        0.0000        0.0000        1.8693
7.8        39301        39301        26360        0        2.0369        0.0000        0.0000        1.7965
6.98        40665        40665        27196        0        2.0192        0.0000        0.0000        1.5833
6.2        42394        42394        28243        0        1.9958        0.0000        0.0000        1.5352
5.46        44018        44018        29211        0        1.9728        0.0000        0.0000        1.4756
4.77        45553        45553        30168        0        1.9609        0.0000        0.0000        1.6557
4.13        47020        47020        31081        0        1.9500        0.0000        0.0000        1.6480
3.54        48572        48572        32063        0        1.9422        0.0000        0.0000        1.7228
3.01        50463        50463        33139        0        1.9129        0.0000        0.0000        1.3202
```

You can also get some quick stats with some linux one-liners on this page; there are more thorough analysis programs built to work with vcf's.

Summary stats - indels and het vs. hom. alleles

```
login1$ grep -c INDEL test.raw.vcf
3432
login1$ cat test.raw.vcf | awk 'BEGIN {FS=";"} {for (i=1;i<=NF;i++) {if (index($i,"AF1")!=0) {print $i} }}'
| \
awk 'BEGIN {FS="="} {print int($2*10)/10}' | sort | uniq -c | sort -n -r | head
  36757 1
  19404 0.5
   1819 0.4
     36 0.6
     17 0.8
     16 0.7
      1 0
```

Keep in mind that variant files only record variation that can be seen with the data provided. Where ever sample sequence exactly matches the reference (i. e. is homozygous wildtype relative to the reference) there will be no data. Which looks the same as if you had **no** data in those regions; this leads us to our next topic.

Multiple-sample variant calling with samtools

This is all fine, but if you're actually trying to study human (or other organism) genetics, you must discriminate homozygous WT from a lack of data. This is done by providing many samples to the variant caller simultaneously. This concept extends further to populations; calling variants across a large and diverse population provides a stronger Bayesian prior probability distribution for more sensitive detection.

To instruct samtools to call variants across many samples, you must simply give it mapped data with each sample tagged separately. Samtools allows two methods to do this:

1. By providing separate bam files for each sample, like this:

   samtools multi-sample variants: separate bam files

   ```
   samtools mpileup -uf hs37d5.fa \
     NA12878.chrom20.ILLUMINA.bwa.CEU.exome.20111114.bam \
     NA12891.chrom20.ILLUMINA.bwa.CEU.exome.20111114.bam \
     NA12892.chrom20.ILLUMINA.bwa.CEU.exome.20111114.bam \
       | bcftools view -vcg - > all.samtools.vcf
   ```

2. By providing one or more bam files, each containing mapped reads from multiple samples tagged with unique samtools `@RG` tags.

   samtools multi-sample variants: one or more bam files using @RG

   ```
   samtools mpileup -uf hs37d5.fa all.bam | bcftools view -vcg - > all.raw.vcf
   ```

The output file from the first option is in `$BI/ngs_course/human_variation`

CAVEAT: If you intend to use the second option, you must make sure you tag your reads with the right RG tag; this can easily be done during the `samse` or `sampe` stage of mapping with bwa with the `-r` option, using `samtools merge`, with picard tools AddOrReplaceReadGroup command, or with your own perl/python/bash commands. Note that our `align_bwa.sh` script takes care of this detail for us.

Filtering

On any variant caller you use, there will be an inherent trade-off between sensitivity and specificity. Typically, you would carry forward as much data as practical at each processing step, deferring final judgement until later so you have more information. For example, you might not want to exclude low coverage regions in one sample from your raw data because you may be able to infer a genotype based on family information later.

This typically leads to NGS pipelines that maximize sensitivity, leading to a substantial number of false positives. Filtering after variant calling can be useful to eliminate false positives based on all the data available after numerous analyses.

In the samtools/bcftools world, the `vcfutils.pl` script provides a means to filter SNPs on many criteria.

Exercises

Explore some filter settings for `vcfutils.pl varFilter` to see how many SNPs get filtered out, using the linux tool `xargs` to do a parameter sweep.

Filtering, counting, and parameter iteration

```
# First - create a tiny shell script to run vcfutils and take a single parameter:
echo "#\!/bin/bash" > tiny.sh
echo "echo \"Sweeping with vcfutils.pl, min read depth of: \$1\" " >> tiny.sh
echo "vcfutils.pl varFilter -Q 20 -d \$1 test.raw.vcf | grep -v '^#' | wc -l " >> tiny.sh

# Now make it executable
chmod +x tiny.sh

# Use xargs to do a lovely sweep
echo 2 3 4 5 6 7 | xargs -n 1 tiny.sh
```

```
Sweeping with vcfutils.pl, min read depth of: 2
41443
Sweeping with vcfutils.pl, min read depth of: 3
32652
Sweeping with vcfutils.pl, min read depth of: 4
22861
Sweeping with vcfutils.pl, min read depth of: 5
14605
Sweeping with vcfutils.pl, min read depth of: 6
9112
Sweeping with vcfutils.pl, min read depth of: 7
5809
```

Exercise

Try modifying tiny.sh on your own to sweep through other filter parameters.

Exercise

Use bedtools and some linux built-in commands to compare the single-sample vcf file(s) to the multiple-sample vcf file (you might need `module load bedtools` unless you've installed that in your `.profile`). Do the following:

1. Count the total number of variants called in `test.raw.vcf` and `all.samtools.vcf`
2. Count how many of the SNPS in these two files are common
3. Calculate the average and maximum quality values for the SNPs that are in test.raw.vcf but NOT in all.samtools.vcf
4. Calculate the average and maximum quality values for the SNPs that are in test.raw.vcf

    1. Investigate `grep -c`
    2. Try `intersectBed` and `wc`
    3. Try `subtractBed` and some `awk`
       Note that for intersections and subtractions on structured data like this, you can use the linux `join` command too.

Comparison of single- and multiple-sample vcf files using linux and bedtools

```
# This command just counts the # of called variants in test.raw.vcf (from individual NA12878)
grep -c -v '^#' test.raw.vcf

# This command just counts the # of called variants in all 3 individuals
grep -c -v '^#' $BI/ngs_course/human_variation/all.samtools.vcf

# Found out how many are common between the two
intersectBed -a test.raw.vcf -b $BI/ngs_course/human_variation/all.samtools.vcf | wc -l

# Take all those that are not in all.samtools.vcf and examine their quality (in field 6 of the vcf file)
subtractBed -a test.raw.vcf -b $BI/ngs_course/human_variation/all.samtools.vcf | \
   awk 'BEGIN {max=0} {sum+=$6; if ($6>max) {max=$6}} END {print "Average qual: "sum/NR "\tMax qual: " max}'

# Look at all the qualities from the NA12878 variants
grep -v '^#' test.raw.vcf | awk 'BEGIN {max=0} {sum+=$6; if ($6>max) {max=$6}} END {print "Average qual: "sum
/NR "\tMax qual: " max}'
```

Output of comparison of single- and multiple-sample vcf files

```
login2$ # This command just counts the # of called variants in test.raw.vcf (from individual NA12878)
login2$ grep -c -v '^#' test.raw.vcf
58049
login2$
login2$ # This command just counts the # of called variants in all 3 individuals
login2$ grep -c -v '^#' $BI/ngs_course/human_variation/all.samtools.vcf
80972
login2$
login2$ # Found out how many are common between the two
login2$ intersectBed -a test.raw.vcf -b $BI/ngs_course/human_variation/all.samtools.vcf | wc -l
52039
login2$
login2$ # Take all those that are not in all.samtools.vcf and examine their quality (in field 6 of the vcf
file)
login2$ subtractBed -a test.raw.vcf -b $BI/ngs_course/human_variation/all.samtools.vcf | \
>    awk 'BEGIN {max=0} {sum+=$6; if ($6>max) {max=$6}} END {print "Average qual: "sum/NR "\tMax qual: "
max}'
Average qual: 8.70595        Max qual: 212
login2$
login2$ # Look at all the qualities from the NA12878 variants
login2$ grep -v '^#' test.raw.vcf | awk 'BEGIN {max=0} {sum+=$6; if ($6>max) {max=$6}} END {print "Average
qual: "sum/NR "\tMax qual: " max}'
Average qual: 43.97        Max qual: 225
```

This data shows that most of the variants that were called on sample NA12878 but are not in the multi-sample variant calls have lower quality.

For major bonus points and a great THANK YOU from Scott, compute the mean and standard deviation of the intersected and subtracted SNPs from NA12878 vs all and then perform a t-test to make sure the differences are statistically significant using only linux command line tools (probably in a shell script). Yes, it's probably easier in Python, Perl, or R.

Other notes on bcftools

bcftools has many other sub-commands such as performing association tests and estimating allele frequency spectrums.

You can't get gene-context information from bcftools - you have to shift to variant annotation tools to do that.

Side-note on `samtools mpileup`

If you don't need a variant caller that uses a full Bayesian inference engine with robust filtering, protections, etc. but just want to look for weird base-artifacts within your mapped data, check out `samtools mpileup` output directly.