Human Variant Calling Workshop

Cesky Krumlov

January 8, 2020

Find the Data

Find the data we're going to use for this workshop. It's in your home directory

cd

From there you can go to where the data are:

cd workshop_materials/alignment_methods

Look at what we have:
Ls

There are two subdirectories, human and yeast. This time we're going to use the human data.

cd human

The Human Data

The data we are working with come from the genome in a bottle (GIAB) human donor, Coriell sample NA12878. This was whole genome resequenced on HiSeq-X to a depth of about 30x. We're going to align and compare it to the human genome reference, build hGRCb37, 1000 Genomes version (this is not the most recent build, but the exercises haven't been updated yet, and the regions we're looking at haven't changed). Because the human genome is very large and it takes a long time to do some of these steps, we're only going to work with data from 3 different 1 megabase strectches of the genome on chromosomes 1, 6, and 20. We will see later why I picked these regions.

Optional: Make Data Read Only

Before we start working, you may want to protect yourself against mistakes. You will need to read, but not write to, all of the files in this directory during the exercise. To make sure you don't accidentally delete these data, you can make them read only:

*chmod 444 **

Decide Where to Work

It is generally good practice to make a directory separate from your raw data to run analyses. You don't have to do this, but if you want to, you are on your own to do this. Throughout this exercise, the full paths to files will not be given. The raw data will always be in /home/genomics/workshop_materials/alignment_methods/human (~/workshop_materials/alignment_methods/human). Any files you create will be wherever you put them, and I will refer to them as, e.g., *<Chr1.sam>*. This <name> notation is a common placeholder for a specific named file (or directory) for which you are assumed to know the path/name but which the writer of the documentation could not know the name. **Do not actually put the <> in your commands;** remember that these are file redirects and you will do unfortunate things you did not expect.

Also, **do not copy/paste the commands from this document**. They will not work. You will have to form your own commands. Remember that you can use tab completion to find and confirm files that already exist.

Prepare to Run BWA

When we did the yeast genome alignments, we made the bwa index for the genome. Making the bwa index for the human genome takes several hours, so I have pregenerated the index for you. Note that although this is a long time, in general you will be able to either get an existing indexed genome (which I did for this) or you will do the indexing once and never do it again.

There is one additional thing we need to do for variant calling: add read groups. Do this:

cat ~/workshop_materials/alignment_methods/human/readgroup.txt

You will see a string:
@RG\tID:NA12878_ALL\tSM:NA12878\tLB:NA12878_ALL\tPL:Illumina\tPU:Multiple\tPM:HiSeq-X\tCN:NYGC

Copy this from your screen (NOT from here, as it will be misformatted) and paste it as shown to form the command on the next page (you can type it if you want, but it is hard to follow.

Run BWA

Now we can run bwa. We're going to use the "mem" algorithm to do the alignments:

*bwa mem –R *
*"@RG\tID:NA12878_ALL\tSM:NA12878\tLB:NA12878_ALL\tPL:Illumina\tPU:Multiple\tPM:HiSeq-X\tCN:NYGC" *
–t 4 human_g1kv37.fa chr1reg.1.fq chr2reg.fq > <chr1>.sam

Note that I haven't included the full paths here, just the file names. You'll have to construct the paths to input files if you are not working in the "human" directory and to the output file if you are. The command goes all on one line, but I have broken it over several lines for readability. If you type the "\" characters, you can also type it over multiple lines, or you can leave out the "\" and type it on one line.

This should take about 3-5 minutes, but see the next page if it's taking longer.

Why Might BWA Take Longer?

We ran into a small snag while testing this. When AWS loads your data onto instance volumes, it doesn't actually pull it all from S3 when you build the instance. It waits until you need the data to pull it down. This means the first time you read a file, it takes a bit longer (maybe twice as long) to read it. Normally we don't even notice this. However, when bwa starts up, the first thing it does it randomly access a very large index. If that index has not been physically moved to your instance, since bwa needs the whole thing but only reads a tiny fraction of it, this will take 50 times or so longer than it normally would, so instead of starting to align in 30 seconds, your bwa may take 10-20 minutes to start aligning. It should be faster on the subsequent runs. Since your instances have all been up for a while, all your data may already be present, and you may not see this at all. Also, since the limiting step is reading the reference indices, which are the same for all three runs, your second two runs should be fast even if your first is slow.

Align the Other Data

After you have aligned the chromosome 1 reads, align the chromosome 6 and 20 reads also. If you do this by editing your previous command line, make sure you change all the filenames you need to change, including your output!

The read group data is the same for all 3 runs, so the –R option will be the same all 3 times.

Convert Output to Binary

Our next step is to convert the sam file into binary format. There are two programs we can use to do this, GATK and samtools. The samtools syntax is a little easier, but the GATK tools are more comprehensive, and in some cases you might want to use other functions GATK provides that it can do simultaneously (as we will see here). Pick one of these (it doesn't matter which) and run it. If you have time (now or later), you can go back and do the other one with a different output file name.

You can even do some of the 3 files with one and some with the other. It will not matter for the later steps.

Either of these should take about 2 minutes to run. Note the second step which is necessary with samtools!

Option 1: Run Samtools

Using samtools:

samtools sort –o <Illumina.samtools>.bam <chrN>.sam

samtools index <Illumina.samtools>.bam

Option 2: Run GATK

Using GATK:

gatk4 SortSam -I <chrN>.sam -O <Illumina.gatk>.bam -SO coordinate --CREATE_INDEX=true

Merge BAMs with samtools

A common occurrence in real data is that we've aligned reads from several different source, usually different lanes of raw sequencing output, but in our case reads from multiple regions of the genome. If you ran multiple regions, convert your extra files to sorted and indexed bams using either method (in truth, if you want to use samtools, it won't matter here if you index them). Now we're going to merge all these bams together.

Again, we can use either samtools or GATK.

samtools merge MERGEDBAM BAMFILE1 BAMFILE2 … BAMFILEN

Note this time samtools takes the output file as an explicit argument, and it comes first with the input files following (in any order). This should give you one bam that's about as big as your three input bams combined. Samtools also sorts while it does this, so we don't need to sort it, but we do need to index. This should run for about 30-60 seconds.

Merge BAMs with GATK

If you do this with GATK, it looks like this:

gatk4 MergeSamFiles -I BAMFILE1 –I BAMFILE2 … -O MERGEDBAM -SO coordinate --CREATE_INDEX true

Note that you give it the -I parameter multiple times, once for each input file to merge. In this case, they are all used (note that in some programs providing the same option more than once results in only the first or only the last one being processed).

Catching Up

If you have not gotten to this point and want to skip to here, you can grab an already merged and sorted file with all three regions:

cp ~/workshop_materials/alignment/human/answers/3regs.merged.bam <working directory>*

Where the second argument is your working directory (. if you are in it). If you do this, your input argument for the next step will literally be "3regs.merged.bam" rather than whatever file names you were working with.

Mark Duplicates

Once you've merged all your bams together and indexed them, we're going to mark duplicates. We do this because sometimes you get exactly the same read or read pair more than once for technical reasons, but they're not unique looks at your underlying sample, so we only want to use one of the duplicate copies for analysis. Samtools has a duplicate marker, but even the samtools developers say that GATK's is better, so we're going to use that

gatk4 MarkDuplicates -I MERGEDBAM -ODEDUPBAM -MDUPMETRICS --CREATE_INDEX=true

Note that it will write a new bam with no changes other than that the duplicates are marked (in the bit flag). We also need to provide a second file, because Picard will give us a text file detailing some metrics about the duplicate marking.

If you don't get all the way through all the files, don't worry. We'll have a duplicate marked file you can start from for the next part.

Catching Up

If you have not made it through everything else, or if you did anything wrong and your file is not working for the next steps, you can copy a working, duplicate-marked, read-group-containing merged bam:

cp ~/workshop_materials/alignment/human/answers/3regs.rg.b <working directory>*

Where the second argument is your current working directory (. if it is your current directory).

If you do this, your bam for the next step will be 3regs.rg.bam.

Base Quality Recalibration I

GATK's BQSR process runs in two steps, one that computes all the quality probabilities and writes out some metrics and a second one that actually generates a new bam with the recalibrated scores in it. We need to run the computation step first:

gatk4 BaseRecalibrator -R human_g1k_v37.fa -I MARKEDBAM -O RECALFILE --known-sites dbsnp_138.b37.vcf.gz

Where `MARKEDBAM` is the bam you duplicate marked and `RECALFILE` is the name of a file to which the recalibration matrix will be written. This is actually a text file, so you can give it a .txt, .tab, or .tsv extension. The knownSites argument (of which we can list multiple) tells the recalibrator which positions to ignore for error rate calculation because they are known variant positions. The reference and known sites files are in the data directory.

This command will take about 5 minutes to run.

Base Quality Recalibration II

Now we use the GATK function ApplyBSQR to apply the calibration and write out a new bam. If you did not get the recalibration file from the previous step, you can copy one to your directory.

cp ~/workshop_materials/alignment/human/answers/3regs.recal.txt <workdir>

This file is then used as RECALFILE in the following command.

gatk4 ApplyBQSR -I MARKEDBAM -O RECALBAM -bqsr RECALFILE -R human_g1k_v37.fa

Where the MARKEDBAM is the same input bam as the previous step (because we didn't change it), RECALBAM is the new bam file we want to write with recalibrated qualities, and RECALFILE is the text file we output in the last step. This takes about 2 minutes to run. Note that GATK outputs an index for your new bam without even being asked to.

Haplotype Caller

The following is the command for haplotype caller, where RECALBAM is your recalibrated bam file and VCFFILE is the name of your vcf file. It is important that your vcf file end in vcf.

gatk4 HaplotypeCaller -R human_g1k_v37.fa -I RECALBAM -O VCFFILE --dbsnp dbsnp_138.b37.vcf.gz

This is going to take about 15 minutes.

Variant Filtering

We have one last step to do, which is to filter the variants. As we discussed, one way to do this is by variant quality recalibration (VQSR), but we don't have enough variants in this data set to run that (plus it takes a really long time), so we're instead going to use a set of standard parameters that are considered "best practices" for human genomes (in other words, if you actually ran VQSR, chances are good you would get something very much like this). Because this cammand is so long, I've spread it across several lines, but it should be run on a single line.

*gatk4 VariantFiltration -R human_g1k_v37.fa *
*-V VCFFILE *
*-O FILTEREDVCF *
*--filter-expression 'QD < 2.0' --filter-name QDfilter *
*--filter-expression 'MQ < 40.0' --filter-name MQfilter *
*--filter-expression 'FS > 60.0' --filter-name FSfilter *
*--filter-expression 'SOR > 3.0' --filter-name SORfilter *
--filter-expression 'ReadPosRankSum < -8.0' --filter-name RPRSfilter

Variant Filtering

You may get a lot of warnings, but you should get out a new vcf. To see if anything happened, you can try:

grep filter FILTEREDVCF | less

You should see a large number of filtered out variants.

Starting IGV

We will spend the rest of the time looking at alignments. For this we will use a tool call IGV (the Integrative Genomics Viewer). To launch IGV, you should be able to either type igv.sh or use the shortcut on your desktop. It will pop up a new window, so if you launch it from the command line, you can place it in the background to free that terminal window.

This time, we're using the default genome for IGV (hGRCb37 is also known as hg19), so you don't need to load the genome.

Loading Your Data

Start by loading your final recalibrated BAM file (if you want, you can use IGV tools to make a coverage track first like we did for the yeast data). If you did not get to a final bam, you can use the one from answers:

~/workshop_materials/alignment/human/answers/3regs.final.bam

When you go to look at IGV, remember we only have 3 windows with reads in them. They are: chr1: 246,155,572-247,155,572 chr6: 157,943,731-158,943,731 chr20: 61,723,571-62,723,571 You can type these into the IGV location window to zoom there, but then you'll have to zoom in to <20,000 bp for your reads to appear. You can do this by dragging a window in the position bar or using the + button in the upper right of the IGV screen (or by typing a smaller window into the location bar). As with yeast, there are some reads in other places. For example, look at chr1: 17,300,000-17,302,000

Loading Your VCF

You can then use File->Load to load your final filtered VCF file and it should show up on the same browser. You might be wondering why we are working with these three regions. They have a particularly high (relatively speaking) density of interesting features. To look at these, you can load the following files into IGV by through the IGV menu, but it's a little more tedious.

~/workshop_materials/alignment/human/tracks/UTAH_1463.dnm.vcf

This is very sparse VCF that contains only sites that were Mendelian inconsistencies between this sample and its parents. Take a look at some of these. Are they called in your VCF? Are they filtered out? The majority of these are not real variants. Can you figure out why they were called anyway? There is also a bed file of validated deletions. You can load that from

~/workshop_materials/alignment/human/tracks/NA12878_lumpy_validated.deletions.bed

Explore in IGV

Spend the rest of the time exploring in IGV, and feel free to ask questions. Here is a list of variants you can look at to decide if you think they're real or not:

Chr Position	
1 : 246200022	1 : 246348769
1 : 247031203	1 : 247031205
1 : 247143449	
6 : 158000199	6 : 158508358
6 : 158839131	6 : 158853142
6 : 158853144	
20 : 62012141	20 : 62158799
20 : 62180956	20 : 62273935
20 : 62631923	

PacBio Calls

We didn't get a chance to work with PacBio long read data for human, but you can load a set of indels called from PacBio sequencing of this individual. PacBio has greater sensitivity for larger insertion and deletion events, because the reads are longer, but less precision for small events because the reads are noisier. You can load a bed of (unvalidated) PacBio indel calls from

~/workshop_materials/alignment/human/tracks/PB_indels.bed