Reference genome based sequence variation detection

Hands-on Exercise

Robert Bukowski (bukowski@cornell.edu), Qi Sun (qisun@cornell.edu)

5/25/2011

OBJECTIVE

Given Illumina sequencing data (in FASTQ format) for four maize lines and the reference genome for maize (in FASTA format), call variants in these lines.

Software to be used (it will be provided on your workshop machine, so no need to download):

Program	Reference URL	Purpose
BWA	http://bio-bwa.sourceforge.net/bwa.shtml#4	Align reads to reference genome
samtools	http://samtools.sourceforge.net/samtools.shtml#4	Operate on SAM and BAM files, convert SAM to BAM, call SNPs and INDELs on multiple samples
picard	http://picard.sourceforge.net/command-line-overview.shtml#MergeBamAlignment	Operate on SAM and BAM files, convert SAM to BAM
GATK	http://www.broadinstitute.org/gsa/wiki/index.php/Best_Practice_Variant_Detection_with_the_GATK_v2#	Re-align reads around indels, call SNPs and INDELSs on multiple samples, filter variants
ANNOVAR	http://www.openbioinformatics.org/annovar/	Annotate variants
IGV	http://www.broadinstitute.org/igv/	Visualize alignments on reference genome

SAMPLE DATA

As the sample data we will use Illumina sequencing reads for four maize lines (B73, Ki11, Mo17, and CML103) from the maize Hapmap project (Gore et al. 2009, *Science*. 326:1115-7). The maize reference genome used is Refgen v2.

PREREQUISITES

Basic familiarity with Linux operating system is a plus. Some general-purpose Linux and scripting tips will be provided throughout the exercise.

You need to have an account on iPlant Atmosphere cloud. Your Atmopshere instance has to be up and running with your EBS volume attached and mounted as **/home/<your_id>/workdir**, where **<your_id>** stands for your Atmosphere user name (e.g., **/home/rb299/workdir**). You have to be connected to your Atmosphere instance using the VNC viewer or an SSH client and have at least one terminal window open. Please consult the document http://cbsu.tc.cornell.edu/lab/doc/wrkshp_Atmosphere.pdf to learn how to launch, configure, and use your Atmosphere instance.

GENERAL TIPS

Initial parts of the exercise (genome indexing and read alignment) are lengthy, about 3 hours total. If you find it convenient, after launching such a long a program (say, genome indexing) you can disconnect from your Atmosphere VM and reconnect later. Your VM and your program will be working while you're away.

You can run different parts of the exercise at different times or on different days. Your Atmosphere VM, once launched, will be waiting for you while you are disconnected. However, if you are not planning on using your VM for a few days, please terminate it (remember to properly unmounts and detach your EBS volume!) to free up the resource. When you are ready to resume the exercise, simply launch a brand new VM and attach your EBS volume to it.

Always examine the output files generated at various stages. The log files (i.e., the screen output) and VCF files are just text files which you can open using a text editor. On Linux, you can use **nano**, **gedit,** or **vi** editors. SAM files are also text files, but they may be too large for text editors to handle (other ways to look into these file are suggested in the text).

Examine (also with a text editor) the shell scripts provided. They are good templates which will save you a lot of command typing if you decide to use any of the tools discussed here for your own project.

You can transfer some of the output files and script files to your own computer for future use.

When you are finished with the exercise, terminate your VM.

INITIAL STEPS

Throughout the exercise the directory **/home/<your_id>/workdir** must always be your current directory, so that any files you create are written there. To change directory to **/home/<your_id>/workdir**, run the following two commands:

```
cd

cd workdir
```

Repeat these commands whenever in doubt what your current directory is. In what follows, the directory **/home/<your_id>/workdir** will be referred to simply as **workdir**.

Step 1. Prepare data

In this step, you will copy the data and executables used in the exercises from the network-mounted common directory **/cbsu/CBSUworkshop** to **workdir**. To do this, execute the following commands:

```
cd; cd workdir

cp /cbsu/CBSUworkshop/CBSUworkshop_bin.tgz  .

tar -xzvf CBSUworkshop_bin.tgz

cp /cbsu/CBSUworkshop/CBSUworkshop_data.tgz .

tar -xzvf CBSUworkshop_data.tgz
```

After the operations complete (which may take a few minutes), **workdir** should contain the following files:

Size [bytes]	File Name
2583122424	B73_full-std.fastq
1872038652	CML103_full-std.fastq
3668943102	Mo17_full-std.fastq
1365755484	Ki11_full-std.fastq
2100151487	maize_agp_v2.fasta
1388	chr10.intervals
	zm2 (directory)
50374154	ZmB73_5a_WGS_chr.gtf

The four ***.fastq** files are Illumina read files for four maize lines, the **maize_agp_v2.fasta** file is the maize reference genome, the **chr10.intervals** file specifies chromosome 10 for GATK, the **gtf** file contains genome annotations for the IGV browser, and the directory **zm2** contains annotation data in UCSC format to be used in the variant annotation step of the exercise. You should also see a few other directories **(samtools, picard, GATK, IGV_15.38, bwa, annovar, scripts**) containing programs used in the exercise. Please verify (using the **ls –al** command) that sizes of the files in your **workdir** are as listed in the table above. If everything looks fine, you can remove the tgz archive files from **workdir** (with the command **rm *.tgz**).

Step 2. Index reference genome

Before the reads can be aligned to the reference genome using BWA, the reference has to be indexed. In **workdir**, run the following (please note that each command has to be typed in one line):

```
mkdir index

./bwa/bwa index -p index/maize_agp_v2 -a bwtsw
maize_agp_v2.fasta  >& index.log  &
```

Remarks:

- Several files will be created in subdirectory **index** with names staring with **maize_agp_v2.**
- The screen output from the command will be captured in the file **index.log**.
- The **&** character at the end of the second command will make it run in the background, so that your terminal will be available for other work.
- Verify that the program is running. Run **top** – you should find **bwa** on top of the list, consuming close to 100% of CPU (to exit top, press Cntrl-C). Run command **ps –ef | grep bwa**. You should see information about your **bwa** process. If you don't – it is not running.
- You can check on the progress of the program by listing the directory index (**ls –al ./index**), or peek into the log file (**more index.log**).
- The indexing will take about one hour.

Step 3. Align reads to reference

For each of the four ***.fastq** files, execute the following set of commands (please note: each command has to be typed in one line)

```
./bwa/bwa  aln -n 2 -t 4 ./index/maize_agp_v2  Ki11_full-std.fastq
1>  Ki11_full-std.sai  2 >> bwa.log  &

./bwa/bwa samse ./index/maize_agp_v2 Ki11_full-std.sai
Ki11_full-std.fastq 1> Ki11_full-std.sam  2>>  bwa.log  &

rm  Ki11_full-std.sai
```

Remarks:

- The first command performs the actual alignment of reads from file **Ki11_full-std.fastq** with up to 2 mismatches allowed per read (**-n 2**).
- It will use all 4 CPU cores available on the machine (**-t 4**).
- The second command converts the alignment result in **sai** format to the SAM format. After this conversion, the intermediate file **Ki11_full-std.sai** may be removed (the third command).
- The three commands above need to be repeated for each of the fastq read files (with **Ki11_full-std** replaced by the appropriate file name core).
- The file **bwa.log** will contain the combined screen output from all commands. Expected combined timing of alignment of all four read files is about 120 minutes.
- **Note**: The '**bwa samse**' command can be launched only after the corresponding '**bwa aln**' command finishes (otherwise, the **sai** file needed by '**bwa samse**' would not be ready). Likewise, the '**sai**' file cannot be removed until '**bwa samse**' is completed.
- How to check the program finished? Run **top** and/or **ps –ef | grep bwa** (see Remarks to Step 2).

As you see, running the alignment involves a lot of typing, especially if you realize that the set of three commands above have to be repeated for each of the four samples (for each of the four fastq files). Thus, it makes sense to invest some time into writing a shell script that could automate things somewhat. For your convenience, we wrote such a simple script called **bwa_aln.sh** – you will find it in **/home/<your_id>/workdir/scripts**. Please open the file **bwa_aln.sh** in your favorite text editor (e.g., **nano, gedit,** or **vi**) and examine its structure. Here are the highlights:

- The first line tells Linux to execute the script using the Bourne Shell (/bin/bash) as an interpreter.
- The sample name, e.g., 'Ki11_full-std', is passed as a command line argument ($1) and stored in variable SAMPLE used in all subsequent commands. This makes it easy to re-use the script for different samples – just supply the sample name as an argument. No changes to the script are necessary.
- Command lines are broken (instead of being typed on a single line) – this is permissible as long as each "piece" ends with the `\' character (backslash). Breaking down the command lines is not necessary - we did this for convenience only (it is just easier to look at various options).
- There is no **&** at the end of any command inside the script (otherwise commands would attempt to start simultaneously, which we do not want) and no output redirection – these will be supplied when the whole script is run (see below).
- There is timing information printed (command **date**) in the beginning and at the end of the script – from this we can tell (looking into bwa.log file) how long it took to run the script.

Thus, instead of laboriously typing all three commands above, we can achieve the same effect by invoking the script (while in **workdir**)

```
./scripts/bwa_aln.sh  Ki11_full-std >>bwa.log 2>&1 &
```

Note the output redirection (2>&1 means that the ‘standard error’ will be written to the same place as ‘standard output’, i.e., to **the baw.log** file) and the **&** at the end means the whole script will be run in the background, one command at a time. To repeat the calculation for a different sample, replace **Ki11_full-std** with that sample’s name. You have to do it three more times.

The <u>SAM files</u> obtained in steps 1-3 are the starting point for SNP/INDEL calling pipelines. Here we will illustrate two such pipelines: one based on the samtools package, and the other – using the Picard and GATK packages.

SNP/INDEL CALLING USING SAMTOOLS

Step 1. SAM to BAM conversion

For each of the SAM files, run the following set of commands (please note: each command has to be typed in one line):

```
./samtools/samtools view -bS -o Ki11_full-std_unsrt.bam
Ki11_full-std.sam  2>>samtools.log  &

./samtools/samtools sort Ki11_full-std_unsrt.bam  Ki11_full-std
2>>samtools.log   &

rm Ki11_full-std_unsrt.bam

./samtools/samtools index Ki11_full-std.bam  2>>samtools.log   &
```

Remarks:

- The initial BAM file **Ki11_full-std_unsrt.bam** produced by the first command needs to be sorted over the genome coordinate (second command). After the sorting step, the initial (unsorted) BAM file is no longer needed and may be removed. Finally, the sorted BAM file is indexed using the fourth command, which produces the index file **Ki11_full-std.bai**.
- The four commands above need to be repeated for each of the remaining SAM files (with **Ki11_full-std** replaced by the appropriate file name core).
- The files **samtools.log** will contain the combined screen output from all samtools commands.
- Expected combined timing for conversion of all four SAM files is about 1 hour.
- **Note**: For each SAM file, the four commands above need to be run one after another, i.e., you should not launch the second command before the first one finishes, etc.

For your convenience, the commands above have been included in script which can be executed simply as:

```
./scripts/Sam2Bam_samtools.sh Ki11_full-std >>samtools.log 2>&1 &
```

sample (different SAM file), replace **Ki11_full_std** in the command above with that sample's name.

Using commands **ls –al *.sam** and **ls –al *.bam**, list the SAM and BAM files you just generated in directory **workdir**. Which ones are larger?

Look into a SAM file, for example, using the command **more Ki11_full-std.sam** (press SPACE to go to next page or 'q' to quit). Identify the header lines and the actual alignment record lines. Consult the SAM format specification at http://samtools.sourceforge.net/SAM1.pdf and identify fields in your actual SAM file. Examine column 12 (OPT) of the SAM file and identify optional tags specific for BWA (http://bio-bwa.sourceforge.net/bwa.shtml#4).

You can also look into a BAM file, although since the file is binary, simple Linux commands will not be enough - you'll need to use a samtools:

```
./samtools/samtools  view  -h  Ki11_full-std.bam  | more
```

(again, use SPACE to go to next page and 'q' to quit). Does it look the same as the SAM file you just examined?

You can visualize the alignments recorded in your BAM files in the context of the reference genome. For details, please read section **VISUALIZING ALIGNMENTS WITH IGV VIEWER** further in this document.

Step 2. Call SNPs and INDELs

Run the following commands (please note: each command has to be typed in one line)

```
./samtools/samtools  mpileup -r chr10 -ugf maize_agp_v2.fasta
Ki11_full-std.bam  CML103_full-std.bam  B73_full-std.bam
Mo17_full-std.bam  | ./samtools/bcftools  view -bcvg  -
1> tmp_samtools.bcf   2>> samtools.log  &

./samtools/bcftools  view tmp_samtools.bcf
|  ./samtools/vcfutils.pl varFilter -D 100
1>  chr10_samtools.vcf  2>> samtools.log &

rm tmp_samtools.bcf
```

Remarks:

- There are actually three sub-steps in this block. '**samtools mpileup**' is used to project the whole depth of sequencing reads from all samples to the each nucleotide of the reference genome and to compute likelihood of data given each possible genotype. Rather than being written to disk, the output in bcf format is piped (see the '**|**' character in the first command) directly to **bcftools** which performs variant calling. In the second command, **vcfutils.pl** is used to filter the results and write them out in **VCF** format (here: to file **chr10_samtools.vcf**).
- In the '**samtools mpileup**' command, option **–r** can be used to select the region of interest. Here we call variants only for chromosome 10 (**–r chr10**). To look for variants over the whole genome, simply remove option **–r chr10** from the first command (it will take significantly more time!)
- Each BAM file specified in the first command is assumed to correspond to one individual (maize line). It is possible to use BAM files containing mixed reads for multiple individuals – the information is then inferred from the read group (RG) tag which must then be present in each record (for details, see 'samtools pileup' portion of http://samtools.sourceforge.net/samtools.shtml#4).
- The VCF file can be open and analyzed in Excel (on your own computer).
- The whole operation should take about 5 minutes.

For your convenience, the commands above have been included in script which can be executed simply as:

```
./scripts/call_snp_indel_samtools.sh  >>samtools.log  2>&1  &
```

(please see comments about scripting in INITAIL STEPS/Step 3). If you wish to repeat the calculations, for example, with different parameters, all you need to do is to edit the file **./scripts/call_snp_indel_samtools.sh**, change the parameters as desired, change the name of the output file, and re-run the script. No need to type everything from scratch.

SNP/INDEL CALLING USING GATK AND PICARD TOOLS

Step 1. SAM to BAM conversion

This step is similar to the analogous step using samtools. GATK developers recommend that Picard toolset is used to manipulate SAM and BAM files. For each of the SAM files, execute the following set of commands (note: each command must be typed in one line):

```
java -jar ./picard/SamFormatConverter.jar INPUT=Ki11_full-std.sam
OUTPUT=Ki11_full-std_noRG.bam  >> gatk.log  2>&1 &

java -jar ./picard/AddOrReplaceReadGroups.jar
INPUT=Ki11_full-std_noRG.bam OUTPUT=Ki11_full-std.bam
SORT_ORDER=coordinate RGID=Ki11_full-std
RGLB=Ki11_full-std  RGPL=solexa  RGSM= Ki11_full-std
RGPU=none  >> gatk.log  2>&1 &

java -jar ./picard/BuildBamIndex.jar
INPUT=Ki11_full-std.bam  >>gatk.log 2>&1  &

rm Ki11_full-std_noRG.bam
```

Remarks:

- GATK requires the BAM files it works with to contain the read group (RG) definitions in the header and the appropriate RG tags in each alignment record. The second command inserts this information into the intermediate BAM file (**Ki11_full-std_noRG.bam**) obtained from the first command. In our simple case, we set read group ID, sample name (SN), and library (LB) to the same value (Ki11_full-std), although in reality, they may be different. See the SAM format specification for more info about read group tag (http://samtools.sourceforge.net/SAM1.pdf).
- The BAM file **Ki11_full-std.bam** resulting from the second command is already sorted over genomic coordinate (see **SORT_ORDER** option), as required by GATK.
- The intermediate file **Ki11_full-std_noRG.bam** can be removed when it is no longer needed.
- The four commands above need to be repeated for all four SAM files (with **Ki11_full-std** replaced by the appropriate file name core).
- The file **gatk.log** will contain the combined screen output from all commands.
- Expected combined timing for conversion of all four SAM files is about 40 minutes.
- **Note**: For each SAM file, the four commands above need to be run one after another, i.e., you should not launch the second command before the first one finishes, etc.

For your convenience, the commands above have been included in script which can be executed simply as:

```
./scripts/Sam2Bam_GATK.sh Ki11_full-std  >>gatk.log  2>&1  &
```

(please see comments about scripting in INITAIL STEPS/Step 3). To repeat the calculation for a different sample (different SAM file), replace **Ki11_full_std** in the command above with that sample's name.

Using commands **ls –al *.sam** and **ls –al *.bam**, list the SAM and BAM files you just generated in directory **/workdir**. Which ones are larger?

Look into a SAM file, for example, using the command **more Ki11_full-std.sam** (press SAPCE to go to next page or 'q' to quit). Identify the header lines and the actual alignment record lines. Consult the SAM format specification at http://samtools.sourceforge.net/SAM1.pdf and compare with your actual file.

You can also look into a BAM file, although since the file is binary, you'll need to use a **samtools** rather than just Linux commands:

```
./samtools/samtools  view  -h  Ki11_full-std.bam  | more
```

(again, use SPACE to go to next page and 'q' to quit). Does it look the same as the SAM file you just examined?

You can visualize the alignments recorded in your BAM files in the context of the reference genome. For details, please read section VISUALIZING ALIGNMENTS WITH IGV VIEWER of this document.

Step 2. Realignment around indels

For each of the BAM files obtained in Step 1, perform re-alignment around indels using the following sequence of commands from the GATK and Picard packages (note: each command must be typed in a single line):

```
java -Xmx2g -jar  ./GATK/GenomeAnalysisTK.jar
-T RealignerTargetCreator -I Ki11_full-std.bam
-R  ./maize_agp_v2.fasta  -L chr10.intervals
-o Ki11_full-std.chr10.targets.interval_list >>gatk.log  2>&1 &

java -Xmx4g -jar ./GATK/GenomeAnalysisTK.jar -T IndelRealigner
-l INFO -I Ki11_full-std.bam  -R  ./maize_agp_v2.fasta
-targetIntervals Ki11_full-std.chr10.targets.interval_list
-L chr10.intervals
-o Ki11_full-std.cleaned_chr10_unsorted.bam  >>gatk.log  2>&1 &

java -jar ./picard/SortSam.jar
INPUT=Ki11_full-std.cleaned_chr10_unsorted.bam
OUTPUT=Ki11_full-std.cleaned_chr10.bam
SORT_ORDER=coordinate >>gatk.log 2>&1 &

rm Ki11_full-std.cleaned_chr10_unsorted.bam

java -jar ./picard/BuildBamIndex.jar
INPUT=Ki11_full-std.cleaned_chr10.bam  >>gatk.log  2>&1  &
```

Remarks:

- The first command looks for candidate loci where re-alignment should be attempted and produces the list of such loci (file **Ki11_full-std.chr10.targets.interval_list**). This list is then passed as input to the actual re-aligner (the second command).
- The BAM file after re-alignment (second command) is not sorted over genomic coordinate. Therefore, it needs to be re-sorted using one of the Picard tools (third command). After sorting, the final BAM file is indexed (fifth command). The intermediate unsorted BAM file is no longer needed and may be removed.
- In this exercise, we only consider a part of the genome, namely – chromosome 10. This is accomplished using the **–L chr10.intervals** option in the first two (GATK) commands. Take a look inside the file **chr10.intervals** – this is an example of an interval list. Interval lists are the way of limiting operation of GATK to certain portions of the genome. Removing this option would make the program consider whole genome (and significantly increase the time of the analysis). To find out more about various formats of interval lists accepted by GATK, see http://www.broadinstitute.org/gsa/wiki/index.php/Input_files_for_the_GATK#Intervals.
- The five commands above have to be repeated for all four samples.
- Screen output generated by all commands will be appended to file **gatk.log**.
- The timing of all re-alignment-related calculations combined over all four samples is about 30 minutes.

For your convenience, the commands above have been included in script which can be executed simply as:

```
./scripts/indel_realign_GATK.sh Ki11_full-std >>gatk.log 2>&1 &
```

(please see comments about scripting in INITAIL STEPS/Step 3). To repeat the calculation for a different sample (different SAM file), replace **Ki11_full_std** in the command above with that sample's name.

Step 3. Call SNPs and INDELs

This step will use all four re-aligned, indexed BAM files sorted over genomic coordinate, obtained in previous steps. The UnifiedGenotyper from GATK will be used to first call the indels, and then SNPs (note that each command must be typed on a single line):

```
java -Xmx6g -jar ./GATK/GenomeAnalysisTK.jar -T UnifiedGenotyper
-nt 4  -glm  INDEL  -L chr10.intervals -R ./maize_agp_v2.fasta
-stand_call_conf 40.0 -stand_emit_conf 20.0 -dcov 200
-A DepthOfCoverage -I Ki11_full-std.cleaned_chr10.bam
-I CML103_full-std.cleaned_chr10.bam
-I B73_full-std.cleaned_chr10.bam
-I Mo17_full-std.cleaned_chr10.bam
-o chr10.indel.vcf  >>gatk.log  2>&1 &

java -Xmx6g -jar ./GATK/GenomeAnalysisTK.jar -T UnifiedGenotyper
-nt 4 -glm SNP -L chr10.intervals  -R maize_ggp_v2.fasta
-stand_call_conf 40.0 -stand_emit_conf 20.0  -dcov 200
-A DepthOfCoverage   -I Ki11_full-std.cleaned_chr10.bam
-I CML103_full-std.cleaned_chr10.bam
-I B73_full-std.cleaned_chr10.bam
-I Mo17_full-std.cleaned_chr10.bam
-o chr10.snp.vcf   >>gatk.log  2>&1 &
```

Remarks:

- Both variant calling steps will use 4 CPU cores (requested by option **-nt 4**). This can be adjusted depending on how many CPU cores are available on a machine.
- **-stand_call_conf** is the minimum phred-scaled Qscore threshold to separate high confidence from low confidence calls. Only genotypes with confidence >= this threshold are emitted as called sites. A reasonable threshold is 30 for high-pass calling (this is the default).
- **-stand_emit_conf** is the minimum phred-scaled Qscore threshold to emit low confidence calls. Genotypes with confidence >= this but less than the calling threshold are emitted but marked as filtered (LowQual). The default value is 30.

For your convenience, the commands above have been included in a script which can be executed simply as:

```
./scripts/call_snp_indel_GATK.sh >>gatk.log  2>&1  &
```

(please see comments about scripting in INITAIL STEPS/Step 3). If you wish to repeat the calculations, for example, with different parameters, all you need to do is to edit the file **./scripts/call_snp_indel_GATK.sh**, change the parameters as desired, change the name of the output file, and re-run the script. No need to type everything from scratch.

Step 4. Filter variants

GATK offers a tool for filtering variants according to a variety of criteria. For details, see http://www.broadinstitute.org/gsa/wiki/index.php/Best_Practice_Variant_Detection_with_the_GATK_v2#Basic_indel_filtering. An example of filtering to try is given below:

```
java -Xmx4g -jar ./GATK/GenomeAnalysisTK.jar -T VariantFiltration
-R ./maize_agp_v2.fasta -filter "MQ0 >= 4 && ((MQ0 / (1.0 * DP)) > 0.1)"
-filter "QUAL<30.0" -filter "SB>-1.0" -filter "QD<2.0"
-filter "AF < 0.05 || AF > 0.95" -filterName HARD_TO_VALIDATE
-filterName IndelQUALFilter -filterName IndelSBFilter
-filterName IndelQDFilter  -filterName IndelAFFilter
-B:variant,VCF chr10.indel.vcf
-o chr10.indel.filtered.vcf >>filter.log  2>&1  &

java -Xmx4g -jar ./GATK/GenomeAnalysisTK.jar -T VariantFiltration
-R ./maize_agp_v2.fasta -B:mask,VCF chr10.indel.filtered.vcf
-filter "MQ0 >= 4 && ((MQ0 / (1.0 * DP)) > 0.1)"  -filter "SB>=0.10"
-filter "QD<5.0" -filter "HRun>=4" -filter "AF < 0.05 || AF > 0.95"
-filterName HARD_TO_VALIDATE  -filterName SNPSBFilter
-filterName SNPQDFilter -filterName SNPHRunFilter -filterName SNPAFFilter
-cluster 3 -window 10  -B:variant,VCF chr10.snp.vcf
-o chr10.snp.filtered.vcf  >>filter.log   2>&1  &
```

Remarks:

- The raw VCF files obtained in previous step are passed as input to **VariantFiltration** using the option **-B:variant,VCF**
- Various filtering criteria based on the INFO field of the input VCF file (see header of the VCF file for definitions of various parameters in the INFO field) are specified using the **–filter** option. In the resulting filtered VCF file (given by the **–o** option) variants failing those filters (i.e., variants satisfying the specified filter criteria) will be marked with labels specified through the corresponding **-filterName** options. For example, variants with allele frequency too low (<0.05) or too high (>0.95) will be marked as SNPAFFilter (in case of SNPs) and IndelAFFilter (for indels) – these criteria are specified using option **-filter "AF < 0.05 || AF > 0.95"** in both commands. Option **-filter "MQ0 >= 4 && ((MQ0 / (1.0 * DP)) > 0.1)"** filters out variants for which reads with zero mapping quality constitute more than 10% of all reads at this site. Variants filtered out with this criterion will be labeled as **HARD_TO_VALIDATE** in the filtered VCF file.

- In SNP filtering, in addition to filters based on the INFO field of the VCF file, two other filters are also applied: for SNPs overlapping with indel variants (specified by option **-B:mask,VCF**), and for SNPs occurring in clusters (3 SNPs within 10bp of one another; options **-cluster 3 -window 10**).

For your convenience, the commands above have been included in script which can be executed simply as:

```
./scripts/var_filter_GATK.sh  >>filter.log   2>&1   &
```

(please see comments about scripting in INITIAL STEPS/Step 3). If you wish to repeat the calculations, for example, with different parameters, all you need to do is to edit the script **./scripts/var_filter_GATK.sh**, change the parameters as desired, change the name of the output file, and re-run the script. No need to type everything from scratch.

ANNOTATE VARIANTS

If you want to annotate the SNP and indels found using any of the variant-calling pipelines, use the ANNOVAR software. For details, see http://www.openbioinformatics.org/annovar/. ANNOVAR requires genomes available through UCSC genome browser. Maize and Arabidopsis genomes are not on the UCSC site, but we have created maize gene annotation files in UCSC format. It is available in the **zm2** directory you copied in the beginning of the exercise.

```
./annovar/convert2annovar.pl chr10.snp.filtered.vcf
-format vcf4 1> chr10.snp.filtered.annovar  2>>annotate.log &

./annovar/annotate_variation.pl  --buildver  zm2
chr10.snp.filtered.annovar ./zm2 1>>annotate.log  2>&1  &
```

For your convenience, the commands above have been included in script which can be executed simply as:

```
./scripts/annotate.sh chr10.snp.filtered  >>annotate.log 2>&1 &
```

(please see comments about scripting in INITIAL STEPS/Step 3). If you wish to run the annotation for a different VCF file (e.g., the indel file), re-run the command above with **chr10.snp.filtered** replaced with the name of that file (excluding the .vcf extension). To repeat the calculations, for example, with different parameters, all you need to do is to edit the script **./scripts/annotate.sh**, change the parameters as desired, and re-run the script. No need to type everything from scratch.

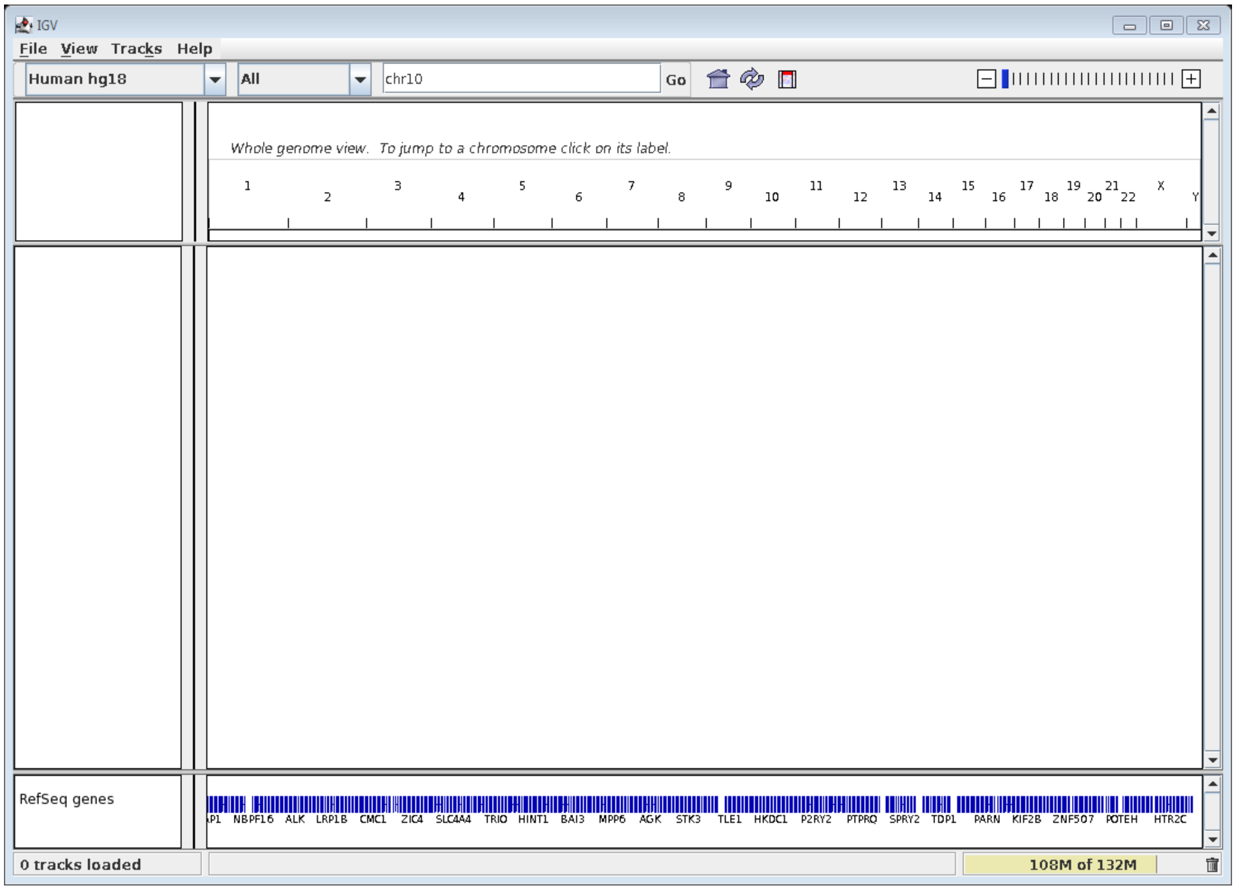
VISUALIZING ALIGNMENTS WITH IGV VIEWER

The read alignments in BAM format can be superimposed on the reference genome and annotation information and visualized using the Integrated Genomics Viewer (IGV), http://www.broadinstitute.org/igv/. A copy of the viewer (Java application) is included in the workshop materials.

To start the IGV viewer:

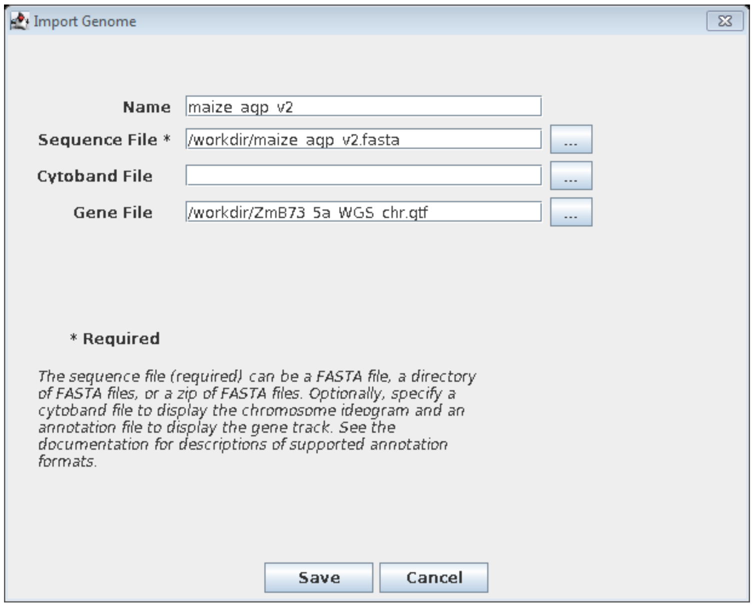
```
cd  IGV_1.5.38

./igv
```

After a few seconds, you should see the graphical user interface preloaded by default with human genome annotations.


IGV
File View Tracks Help
Human hg18
All
chr10
Go
Whole genome view. To jump to a chromosome click on its label.
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 X Y
RefSeq genes
NBPF16 ALK LRP1B CMC1 ZIC4 SLC4A4 TRIO HINT1 BAI3 MPP6 AGK STK3 TLE1 HKDC1 P2RY2 PTPRQ SPRY2 TDP1 PARN KIF2B ZNF507 POTEH HTR2C
0 tracks loaded
108M of 132M

You will need to load the maize genome you used in your alignments. To do this, click on **File->Import Genome**. In the dialog that appears, specify **maize_agp_v2** as Name, **/home/<your_id>/workdir/maize_agp_v2.fasta** as the Sequence File, and the annotation file

/home/<your_id>/workdir/ZmB73_5a_WGS-chr.gtf as the Gene File (you can select the files by browsing to them). Click **Save**, specifying **/home/<your_id>/workdir** as the location of the new IGV-formatted genome file.



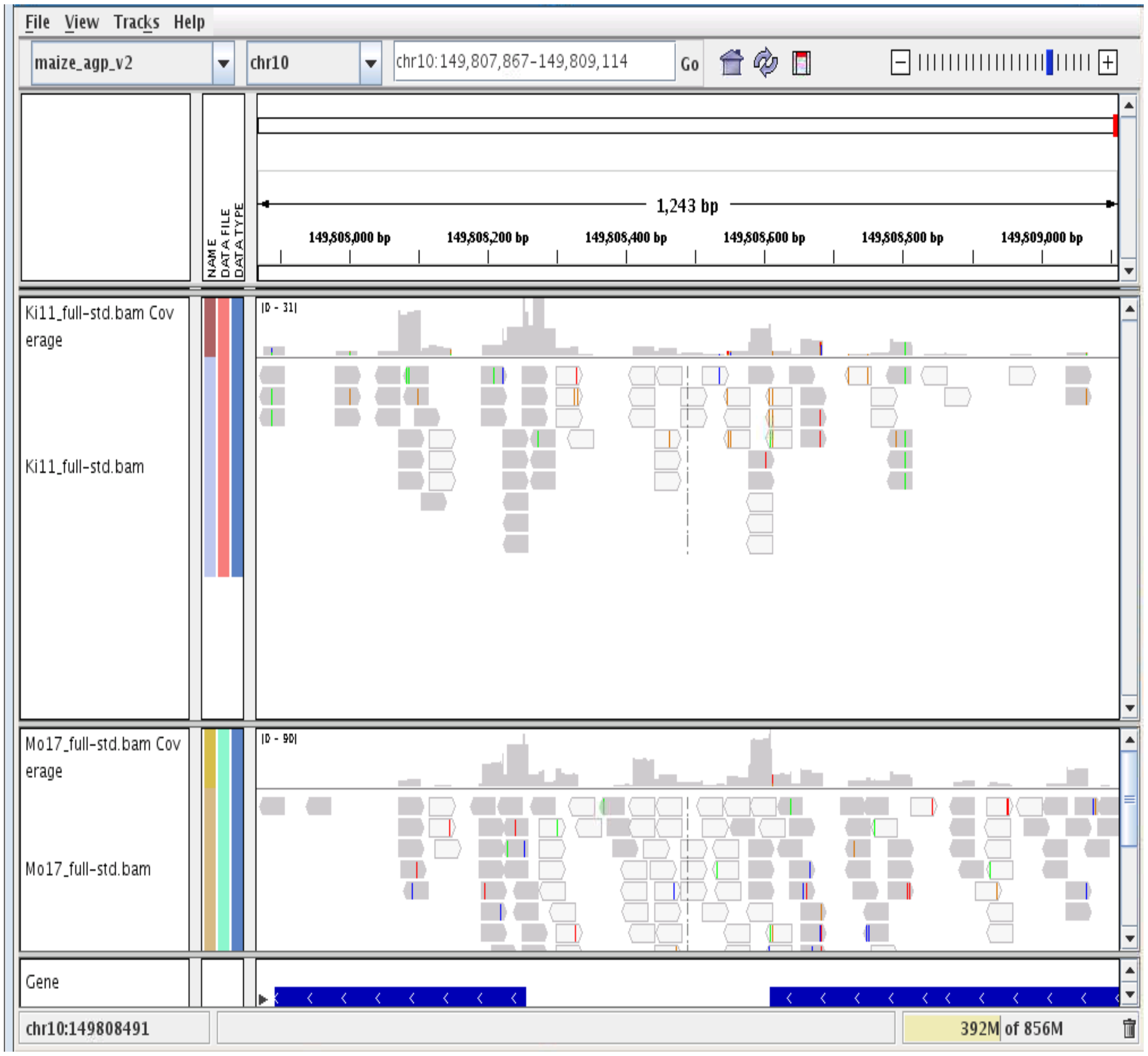
Name: maize_agp_v2
Sequence File *: /workdir/maize_agp_v2.fasta
Cytoband File:
Gene File: /workdir/ZmB73_5a_WGS_chr.gtf

* Required

The sequence file (required) can be a FASTA file, a directory of FASTA files, or a zip of FASTA files. Optionally, specify a cytoband file to display the chromosome ideogram and an annotation file to display the gene track. See the documentation for descriptions of supported annotation formats.

The maize genome is now loaded into the viewer and should show up in the genome selection dropdown (top left corner of the IGV window).

Now load one or more BAM files by clicking on **File->Load From File**, then navigating to **/home/<your_id>/workdir** and selecting a BAM file of your choice. You can load more BAM files repeating this operation. By the nature of the experiment the reads were obtained from, coverage of the genome is very “spotty”, i.e., the reads align to a small number of loci scattered widely over the genome, while most of the genome is not covered at all. Therefore, what you will see right after loading the BAM files will probably resemble blank screen and will not be too exciting .To see anything interesting, you will need to zoom in to a particular chromosome and locus where read coverage is substantial. If you performed the SNP/Indel calling part of the exercise, you can use the VCF files as a guide. Peek into one of these files and select coordinates of one or more variants, then zoom in to the region around these coordinates. For example, if you type **chr10:149,807,867-149,809,114** in the text box on top of the IGV window and click **Go**, you should see something like this:



The screenshot above shows data from two loaded BAM files. The reads are represented by arrows with the direction corresponding to the strand to which a given read aligns. Light gray colored reads have zero mapping quality. Differences with respect to reference are shown as colored vertical lines, where color corresponds to the base.

Use zoom tool (+/- buttons) to change the scale. You can left-click and drag anywhere within the screen to change the coordinate range. Hover anywhere to see more detailed information about the object. Right-click anywhere for more display options.