GATK HaplotypeCaller Analyzing of BWA (mem) mapped Illumina reads

SP©BITS

(October 4th, 2013 - v1.00)

1 Introduction

1.1 Aims

The Broad Institute **GATK** suite is today's high end standard for NGS data analysis. Although a lot of documentation is present on the GATK website, it can still be challenging to apply a full analysis workflow to your own data for the first time.

This document details each steps of a prototype GATK analysis using hg19 mapping data. The data used here was generated using Illumina reads of the HapMap individual **NA18507** as described in 'Hands-On BITS session: *Mapping Illumina reads to the human genome and calling variations on chr21*'

Readers are expected to have done the full analysis using hands-on workflows and have access to some of the data and reference files used there. Additional data is required to use GATK tools including the GATK executable (java) as well as Broads **bundle** files matching the reference genome used in the training as detailed inline.

REM: Please NOTE that GATK can be obtained by academic researchers for research purpose by accessing the Broad platform http://www.broadinstitute.org/gatk/. For profit and industrial users, a special agreement should be made with the Broad Institute prior to getting their code. This training session is meant for non-profit users that are entitled to use the GATK material.

1.2 Resources and Reference links

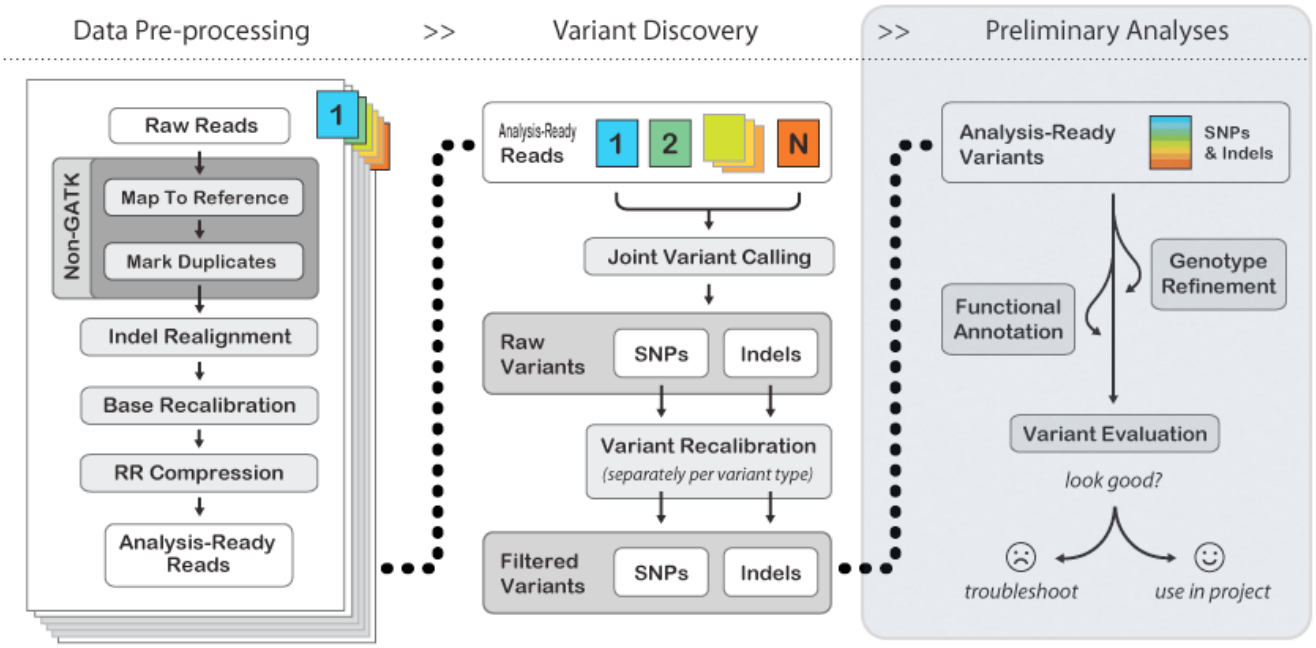
!! Resources linked here might be outdated at the time you read this document.

IMPORTANT NOTE: Most of the information used here was copied from pages organized on the GATK online platform, The broad Institute should therefore receive full credit if you use this material elsewhere. In order to cite GATK, please follow instructions on their dedicated page: http://www.broadinstitute.org/gatk/about/citing-gatk

- **GATK** online help is accessible at http://www.broadinstitute.org/gatk/guide/topic?name=tutorials.
- A PDF version of these pages is available in the folder */gatk/pdfdocs/*. Look for the latest version at the time your access the page. http://www.broadinstitute.org/gatk/pdfdocs/GATK_GuideBook_2.7-2.pdf
- The Java code is available for academic at http://www.broadinstitute.org/gatk/download) and is installed by simply storing the uncompressed folder in the computer PATH.
- The Broad CRAN **gsalib** package is additionally required to generate plots in GATK. The package is found at http://cran.r-project.org/web/packages/gsalib/index.html and can be installed directly from within **RStudio** with the simple command '*install.packages("gsalib")*'. After installation, this package should be accessible for the GATK commands, if this is not the case, please refer to the GATK help and set your environment to make the **gsalib** code part of the standard path. The documentation can be accessed on the CRAN pages http://cran.r-project.org/web/packages/gsalib/gsalib.pdf.
- Finally, Broad **Bundle** files are required for annotating and filtering your data and can be found on the public FTP site. We provide a script to batch download these files in appendix. Additional reference files are also required but were already obtained for BITS NGS training.
- SnpEff used to extract variant subsets but that can do much more. Please refer to the online documentation for the full description. http://snpeff.sourceforge.net/index.html
- vcfTools already used previously to compute VENN diagram from the obtained and reference VCF files. http://vcftools.sourceforge.net.

2 A GATK Best Practice Workflow

The current Best practice for GATK is illustrated in the next figure and constitutes the framework for this tutorial.



The presented workflow is directly inspired from the GATK Best Practice slide reproduced above. Each step was extracted from the corresponding GATK online documentation and adapted to fit the location of our files. The right panel (grey), dedicated to downstream analysis is not part of this tutorial which aims at generating high quality data.

2.1 Defining file path for the workflows

The path is adapted from that of the BITS hands-on NGS training.

```
project=/Users/bits/NGS_DNASeq-training
base=${project}/Final_Results
reference=${project}/ref/HiSeq_UCSC_hg19.fa

mapping=${base}/hg19_mapping-full

# create a folder to receive all results from this GATK workflow
result=${base}/gatk
mkdir -p ${result}
```

2.2 Add some indexing to our reference files (If not already available)

```
# create reference sequence dictionary if not present
[ -f "${project}/ref/HiSeq_UCSC_hg19.dict" ] || \
java -Xmx6g -jar $PICARD/CreateSequenceDictionary.jar \
    R=${reference} \
    O=${project}/ref/HiSeq_UCSC_hg19.dict \
    GENOME_ASSEMBLY=hg19

# create reference index if not present
[ -f "${project}/ref/HiSeq_UCSC_hg19.fa.fai" ] || \
samtools faidx ${project}/ref/HiSeq_UCSC_hg19.fa
```

2.3 Prepare the GATK input data and Workflow elements

2.3.1 Clean the initial BWA bam data to make it Picard & GATK compliant

In this part, we will fix a BAM that is not indexed or not sorted, has not had duplicates marked, or is lacking read group information. These steps can be performed independently of each other but the order proposed here is recommended.

```
# Use the mapping file obtained with BWA (mem)
ori-bam=${base}/hg19_mapping/NA18507_GAIIx_100_chr21_aln-pe.bam
```

2.3.2 Correct potential mate pair info and coordinate Sort the reads (requires Picard tools installed and running).

```
java -Xmx6g -jar $PICARD/FixMateInformation.jar \
    I=${result}/${ori-bam} \
    O=${result}/sorted_reads.bam \
    SO=coordinate \
    VALIDATION_STRINGENCY=LENIENT
```

2.3.3 Mark duplicate reads (optical duplicates could bias variant detection by adding excessive coverage depth at a variant locus.

```
java -Xmx6g -jar $PICARD/MarkDuplicates.jar \
    I=${result}/sorted_reads.bam \
    O=${result}/dedup_reads.bam \
    M=${result}/duplicate_metrics.txt
```

The result is a text file starting by:

```
## net.sf.picard.metrics.StringHeader
# net.sf.picard.sam.MarkDuplicates INPUT=[sorted_reads.bam] OUTPUT=dedup_reads.bam METRICS_FILE=metrics
## net.sf.picard.metrics.StringHeader
# Started on: Thu Oct 03 12:21:20 CEST 2013

## METRICS CLASS        net.sf.picard.sam.DuplicationMetrics
LIBRARY UNPAIRED_READS_EXAMINED READ_PAIRS_EXAMINED     UNMAPPED_READS  UNPAIRED_READ_DUPLICATES
        37296   7379961 192976  8610    3101    1059    0.001001        13329615355

## HISTOGRAM    java.lang.Double
BIN     VALUE
1.0     1.000143
2.0     1.999733
3.0     2.99877
4.0     3.997254
5.0     4.995185
6.0     5.992563
...
```

2.3.4 Add read group information required by GATK (adapted from our BWA command)

```
# ID:NA18507
# LB=lib-NA18507
# PU=unkn-0.0
# PL:ILLUMINA
# SM:GAIIx-chr21-BWA.mem
java -Xmx6g -jar $PICARD/AddOrReplaceReadGroups.jar \
```

```
    INPUT=${result}/dedup_reads.bam \
    OUTPUT=${result}/addrg_reads.bam \
    RGID="NA18507" \
    RGLB="lib-NA18507" \
    RGPL="ILLUMINA" \
    RGPU="unkn-0.0" \
    RGSM="GAIIx-chr21-BWA.mem.gatk"
```

2.3.5 Index the last file for further use

```
java -Xmx6g -jar $PICARD/BuildBamIndex.jar \
    INPUT=${result}/addrg_reads.bam
```

2.3.6 Create symbolic links to reference data required for the GATK workflow

A number of GATK bundle files are necessary to run this workflow. The files for build hg19 were obtained from the GATK ftp site and are defined below. The full bundle can be obtained with the script **wget-bundle.sh** attached in appendix-01

```
# BUNDLE: locate important files for GATK analysis
ver=2.5
bundle=$BIODATA/bundle_${ver}/hg19
# INDEL gold standards
mills=${bundle}/Mills_and_1000G_gold_standard.indels.hg19.vcf.gz
phase1indel=${bundle}/1000G_phase1.indels.hg19.vcf.gz
# pick one of the former two as gold_indels set
gold_indels=${mills}
# SNP gold standards
dbsnp=${bundle}/dbsnp_137.hg19.vcf.gz
hapmap=${bundle}/hapmap_3.3.hg19.vcf.gz
phase1snp=${bundle}/1000G_phase1.snps.high_confidence.hg19.vcf.gz
omni=${bundle}/1000G_omni2.5.hg19.vcf.gz
```

2.4 Improve mapping by local realignment (indels)

2.4.1 Identify regions for indel local realignment of the selected chromosome

```
java -Xmx6g -jar $GATK/GenomeAnalysisTK.jar \
    -T RealignerTargetCreator \
    -R ${reference} \
    -I ${result}/addrg_reads.bam \
    -L chr21 \
    -known ${gold_indels} \
    -o ${result}/target_intervals.list
```

The target_intervals.list starts by:

```
chr21:9442356-9442357
chr21:9466976-9466977
chr21:9467571-9467572
chr21:9467834-9467835
chr21:9470889-9470890
chr21:9474246-9474247
chr21:9476537-9476609
...
```

2.4.2 Perform indel local realignment of the selected chromosome

```
java -Xmx6g -jar $GATK/GenomeAnalysisTK.jar \
    -T IndelRealigner \
    -R ${reference} \
    -I ${result}/addrg_reads.bam \
    -L chr21 \
    -targetIntervals ${result}/target_intervals.list \
    -known ${gold_indels} \
    -o ${result}/realigned_reads.bam
```

2.4.3 Analyze patterns of covariation in the sequence dataset

```
java -Xmx6g -jar $GATK/GenomeAnalysisTK.jar \
    -T BaseRecalibrator \
    -R ${reference} \
    -I ${result}/realigned_reads.bam \
    -L chr21 \
    -knownSites ${dbsnp} \
    -knownSites ${gold_indels} \
    -o ${result}/recal_data.table
```

2.4.4 Do a second pass to analyze covariation remaining after recalibration

Use the same BAM data as in the first iteration

```
java -Xmx6g -jar $GATK/GenomeAnalysisTK.jar \
    -T BaseRecalibrator \
    -R ${reference} \
    -I ${result}/realigned_reads.bam \
    -L chr21 \
    -knownSites ${dbsnp} \
    -knownSites ${gold_indels} \
    -BQSR ${result}/recal_data.table \
    -o ${result}/post_recal_data.table
```

2.4.5 Generate before/after plots

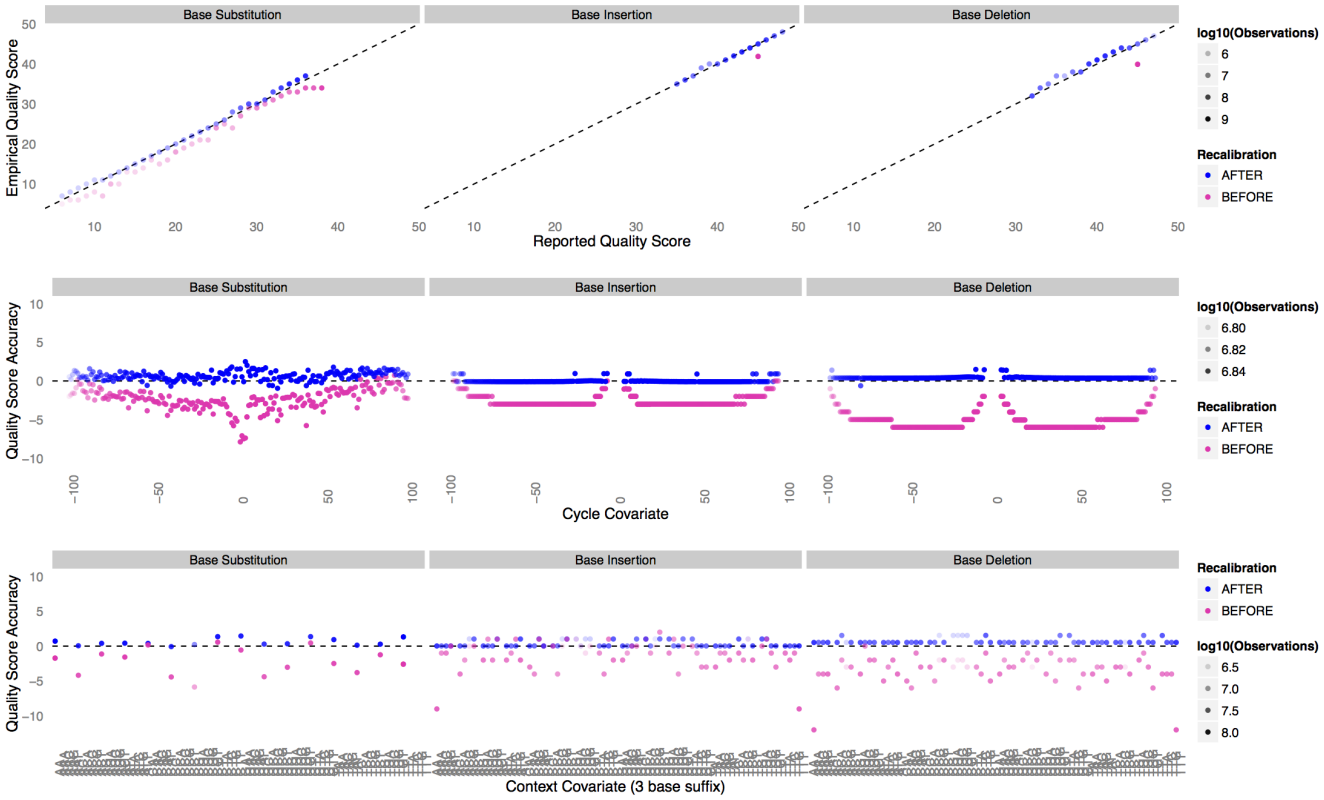
Requires install.packages("gsalib") in RStudio

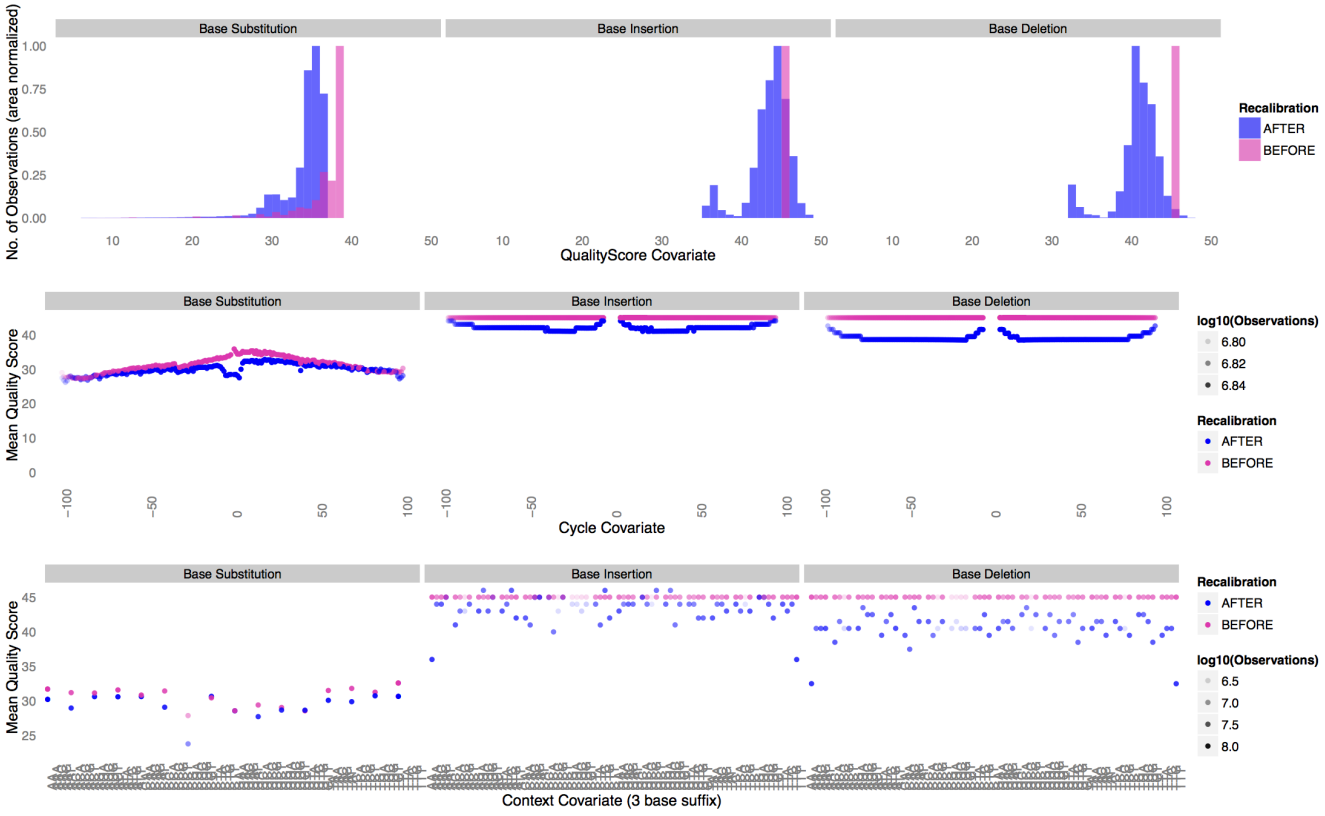
```
java -Xmx6g -jar $GATK/GenomeAnalysisTK.jar \
    -T AnalyzeCovariates \
    -R ${reference} \
    -L chr21 \
    -before ${result}/recal_data.table \
    -after ${result}/post_recal_data.table \
    -plots ${result}/recalibration_plots.pdf
```

GATK BaseRecalibration report

Argument	Value
binary_tag_name	None
covariate	ReadGroup,QualityScore,Context,Cycle
default_platform	None
deletions_default_quality	45
force_platform	None
indels_context_size	3
insertions_default_quality	45
low_quality_tail	2
maximum_cycle_value	500
mismatches_context_size	2
mismatches_default_quality	−1
no_standard_covs	false
quantizing_levels	16
run_without_dbsnp	false
solid_nocall_strategy	THROW_EXCEPTION
solid_recal_mode	SET_Q_ZERO

Thu Oct 3 20:15:17 2013





Overall error rates by event type

ReadGroup	unkn–0.0	unkn–0.0	unkn–0.0
EventType	M	I	D
EmpiricalQuality	29.0	42.0	40.0
EstimatedQReported	30.7	45.0	45.0
Observations	1.41e+09	1.41e+09	1.41e+09
Errors	1.96e+06	8.60e+04	1.56e+05

Error rates by event type and initial quality score

ReadGroup	QualityScore	EventType	EmpiricalQuality	Observations	Errors
unkn–0.0	6	M	5	198442	58252.75
unkn–0.0	7	M	6	360945	96314.31
unkn–0.0	8	M	6	458407	112837.35
unkn–0.0	9	M	7	473933	85634.52
unkn–0.0	10	M	8	546186	82145.47
unkn–0.0	11	M	7	1157813	210744.77
unkn–0.0	12	M	10	2522834	232161.97
unkn–0.0	13	M	10	566603	52837.38
unkn–0.0	14	M	13	349011	16503.83
unkn–0.0	15	M	13	565430	26117.59
unkn–0.0	16	M	14	529656	19608.37
unkn–0.0	17	M	16	766175	21029.05
unkn–0.0	18	M	15	930669	30649.90
unkn–0.0	19	M	16	1306367	29852.30
unkn–0.0	20	M	18	6591149	99141.91
unkn–0.0	21	M	19	1564470	20453.66
unkn–0.0	22	M	20	1394332	14331.62
unkn–0.0	23	M	21	1375247	11243.08
unkn–0.0	24	M	21	1732364	13230.93
unkn–0.0	25	M	24	11703007	51705.73
unkn–0.0	26	M	25	3129306	9288.97
unkn–0.0	27	M	24	2548276	9415.49
unkn–0.0	28	M	27	16394646	32914.82
unkn–0.0	29	M	29	6278461	8068.72
unkn–0.0	30	M	29	25933176	31752.06
unkn–0.0	31	M	30	13191274	11988.86
unkn–0.0	32	M	31	31563269	24850.11
unkn–0.0	33	M	32	46005667	28191.61
unkn–0.0	34	M	33	40975323	21936.46
unkn–0.0	35	M	33	77546972	36648.33
unkn–0.0	36	M	34	199673225	84411.15
unkn–0.0	37	M	34	161993877	64097.03
unkn–0.0	38	M	34	746492184	308085.10
unkn–0.0	45	I	42	1406818696	85965.73
unkn–0.0	45	D	40	1406818696	156072.22

2.4.6 Apply the recalibration to your sequence data

```
java -Xmx6g -jar $GATK/GenomeAnalysisTK.jar \
    -T PrintReads \
    -R ${reference} \
    -L chr21 \
    -I ${result}/realigned_reads.bam \
    -BQSR ${result}/recal_data.table \
    -o ${result}/recal_reads.bam
```

2.4.7 Compress your sequence data

The BAM file obtained in the last step is about twice as big as the initial BAM. IN this step, the size is brought back to a fraction of this initial size (about half). It should also speed-up analysis by other GATK tools.

```
java -Xmx6g -jar $GATK/GenomeAnalysisTK.jar \
    -T ReduceReads \
    -R ${reference} \
    -I ${result}/recal_reads.bam \
    -L chr21 \
    -o ${result}/reduced_reads.bam
```

2.5 Call all variants using the HaplotypeCaller

2.5.1 Call variants in your sequence data (diploid genome => HaplotypeCaller)

```
java -Xmx6g -jar $GATK/GenomeAnalysisTK.jar \
    -T HaplotypeCaller \
    -R ${reference} \
    -I ${result}/reduced_reads.bam \
    -L chr21 \
```

```
    --genotyping_mode DISCOVERY \
    -stand_emit_conf 10 \
    -stand_call_conf 30 \
    -o ${result}/raw_variants.vcf
```

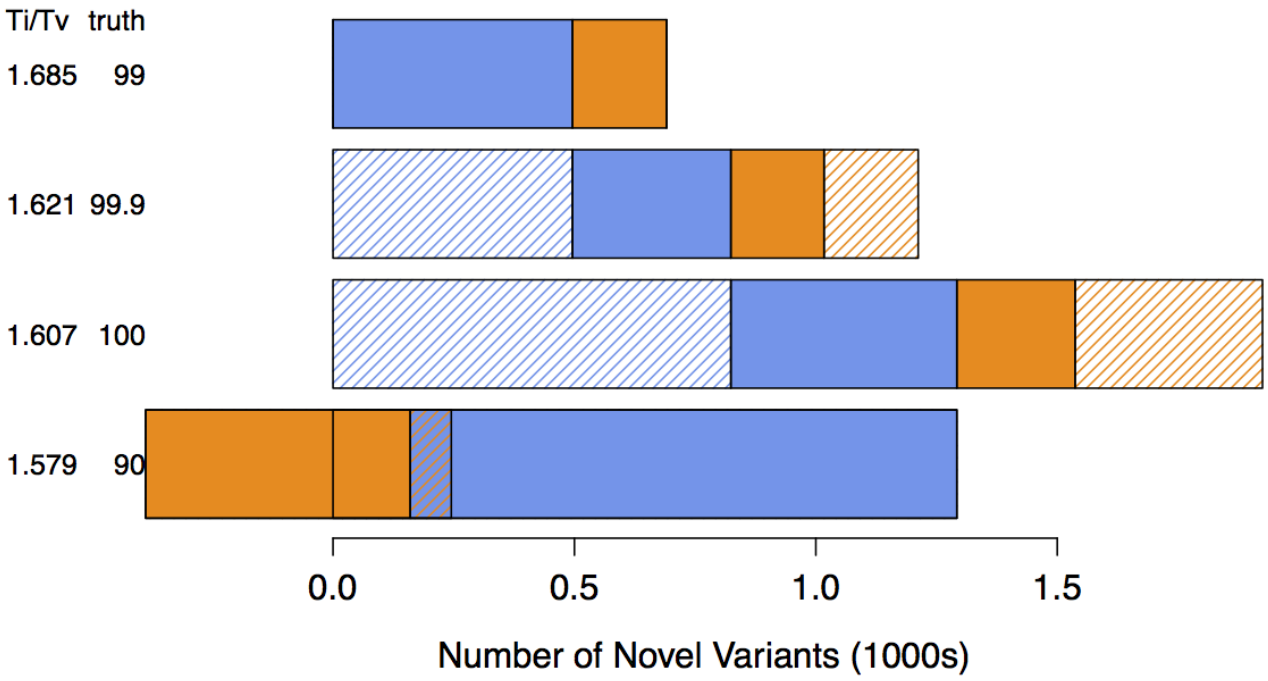
2.5.2 Recalibrate variant scores in two steps: SNP then INDELS

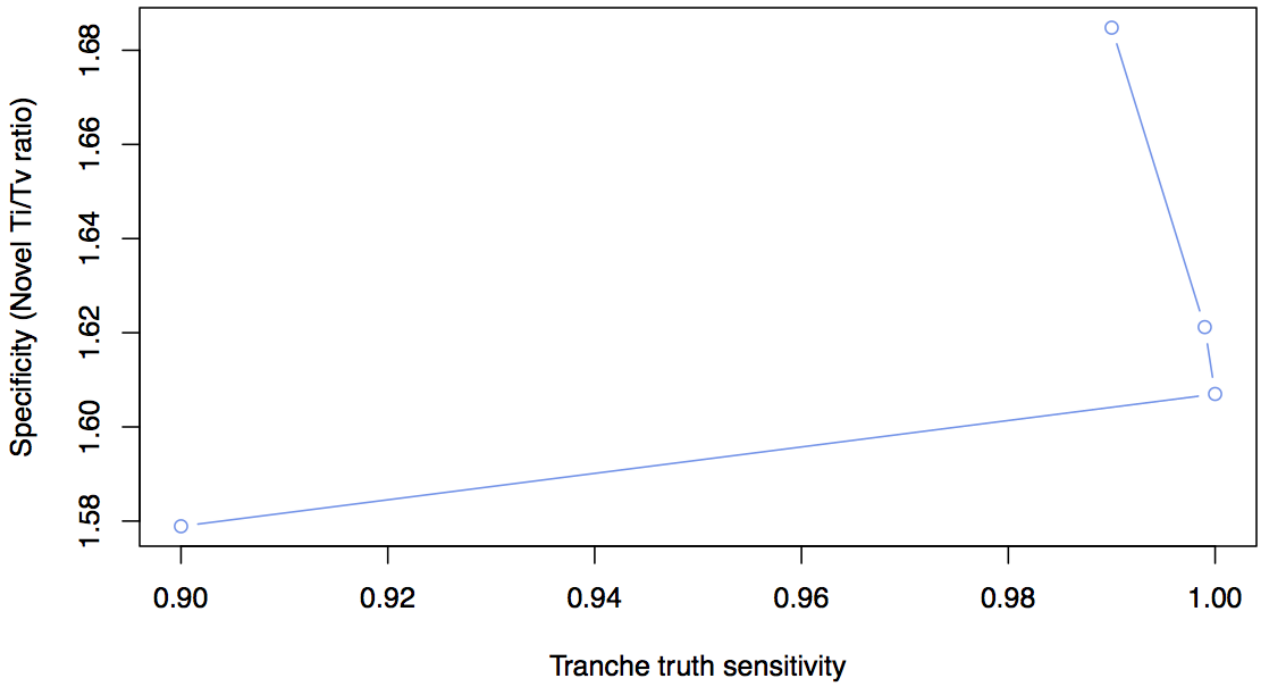
During this step, additional annotations are added to the data that allow later filtering and selection of subsets. Some of the available annotations are listed here.

- Build the SNP recalibration model
- also add annotations from sources out of:
- BaseQualityRankSumTest (BaseQRankSum)
- DepthOfCoverage (DP)
- FisherStrand (FS)
- HaplotypeScore (HaplotypeScore)
- MappingQualityRankSumTest (MQRankSum)
- MappingQualityZero (MQ0)
- QualByDepth (QD)
- ReadPositionRankSumTest (ReadPosRankSum)
- RMSMappingQuality (MQ)
- SnpEff: Add genomic annotations using the third-party tool SnpEff with VariantAnnotator

```
java -Xmx6g -jar $GATK/GenomeAnalysisTK.jar \
    -T VariantRecalibrator \
    -R ${reference} \
    -input ${result}/raw_variants.vcf \
    -resource:hapmap,known=false,training=true,truth=true,prior=15.0 ${hapmap} \
    -resource:omni,known=false,training=true,truth=false,prior=12.0 ${omni} \
    -resource:1000G,known=false,training=true,truth=false,prior=10.0 ${phase1snp} \
    -resource:dbsnp,known=true,training=false,truth=false,prior=2.0 ${dbsnp} \
    -an DP \
    -an QD \
    -an FS \
    -an MQRankSum \
    -an ReadPosRankSum \
    -mode SNP \
    -tranche 100.0 \
    -tranche 99.9 \
    -tranche 99.0 \
    -tranche 90.0 \
    -numBad 1000 \
    -recalFile ${result}/recalibrate_SNP.recal \
    -tranchesFile ${result}/recalibrate_SNP.tranches \
    -rscriptFile ${result}/recalibrate_SNP_plots.R
```

Again, a number of plots are generated by R



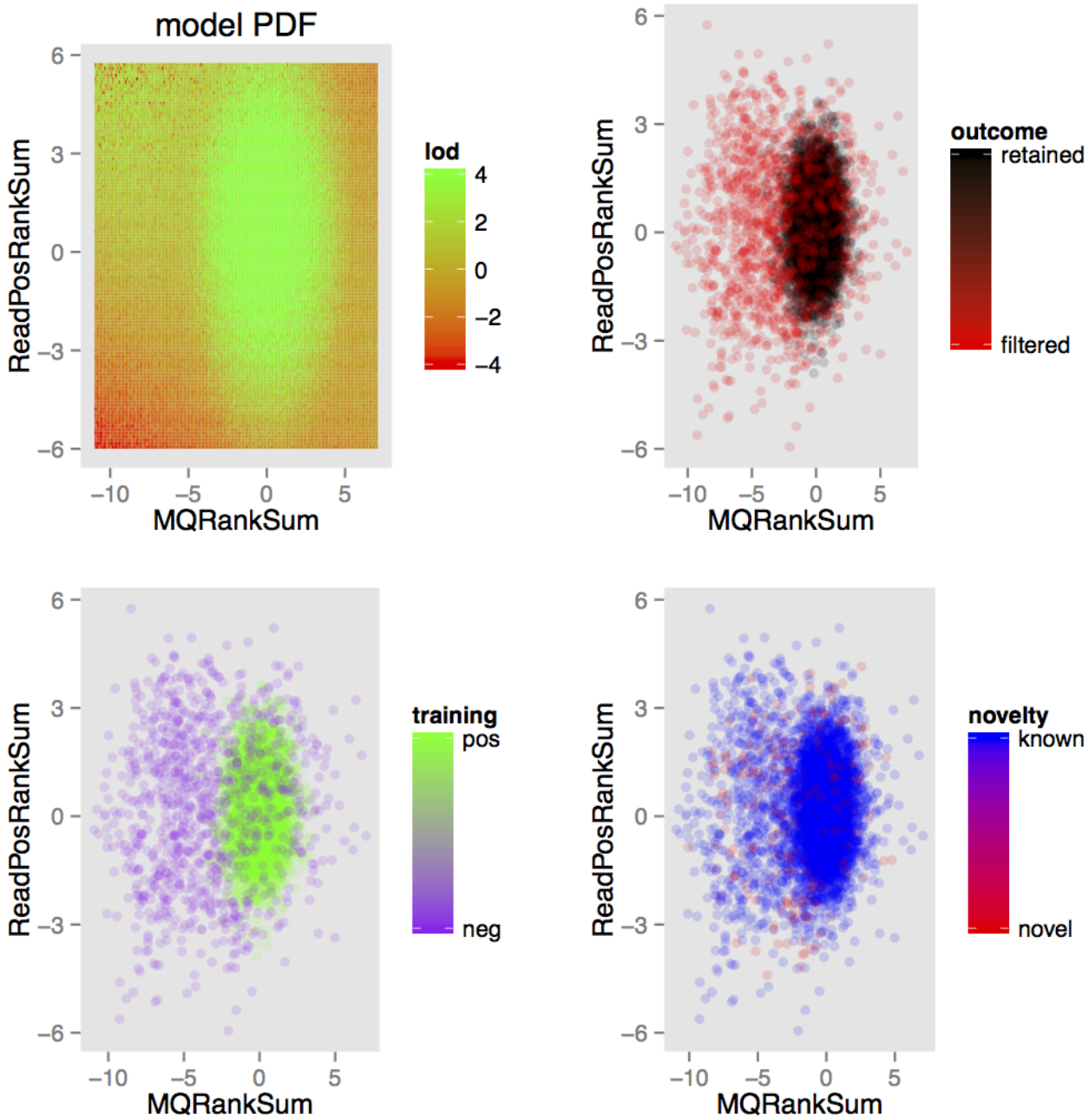


2.5.3 Apply SNP recalibration

```
java -Xmx6g -jar $GATK/GenomeAnalysisTK.jar \
    -T ApplyRecalibration \
    -R ${reference} \
    -input ${result}/raw_variants.vcf \
```

```
    -mode SNP \
    --ts_filter_level 99.0 \
    -recalFile ${result}/recalibrate_SNP.recal \
    -tranchesFile ${result}/recalibrate_SNP.tranches \
    -o ${result}/recalibrated_snps_raw_indels.vcf
```

Detailed plots are provided for each step of the recalibration. We only reproduce here page #1 of 10 similar pannels

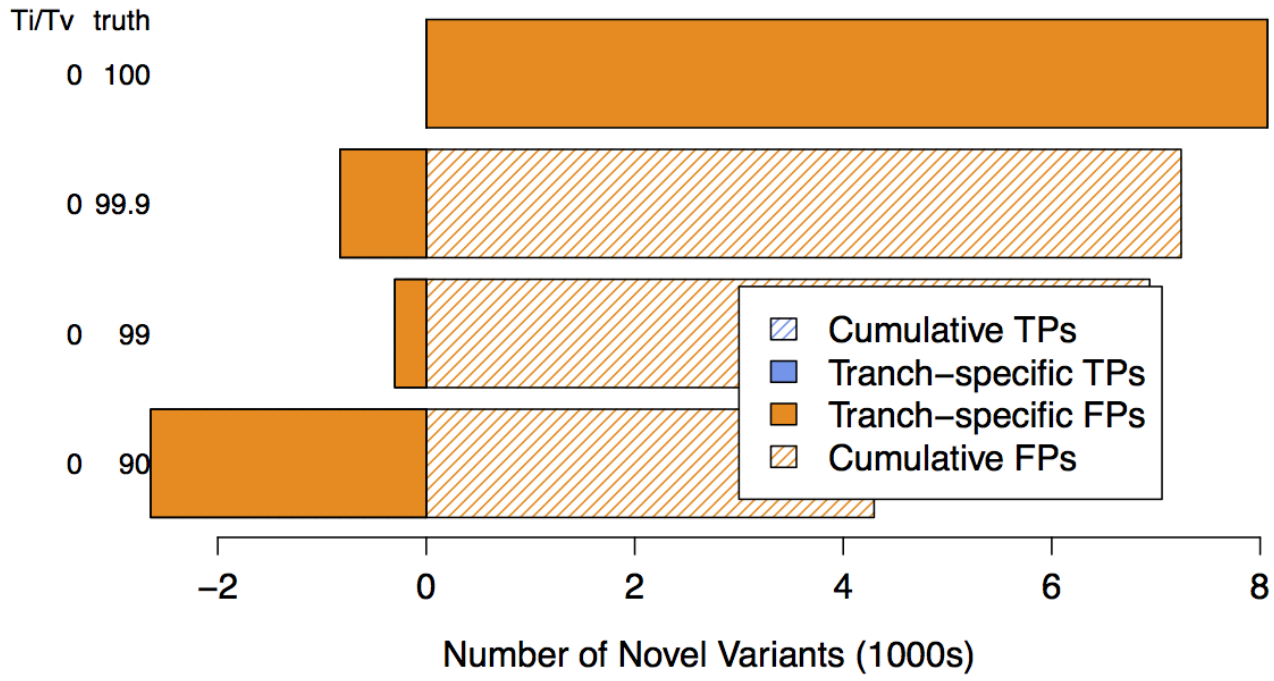


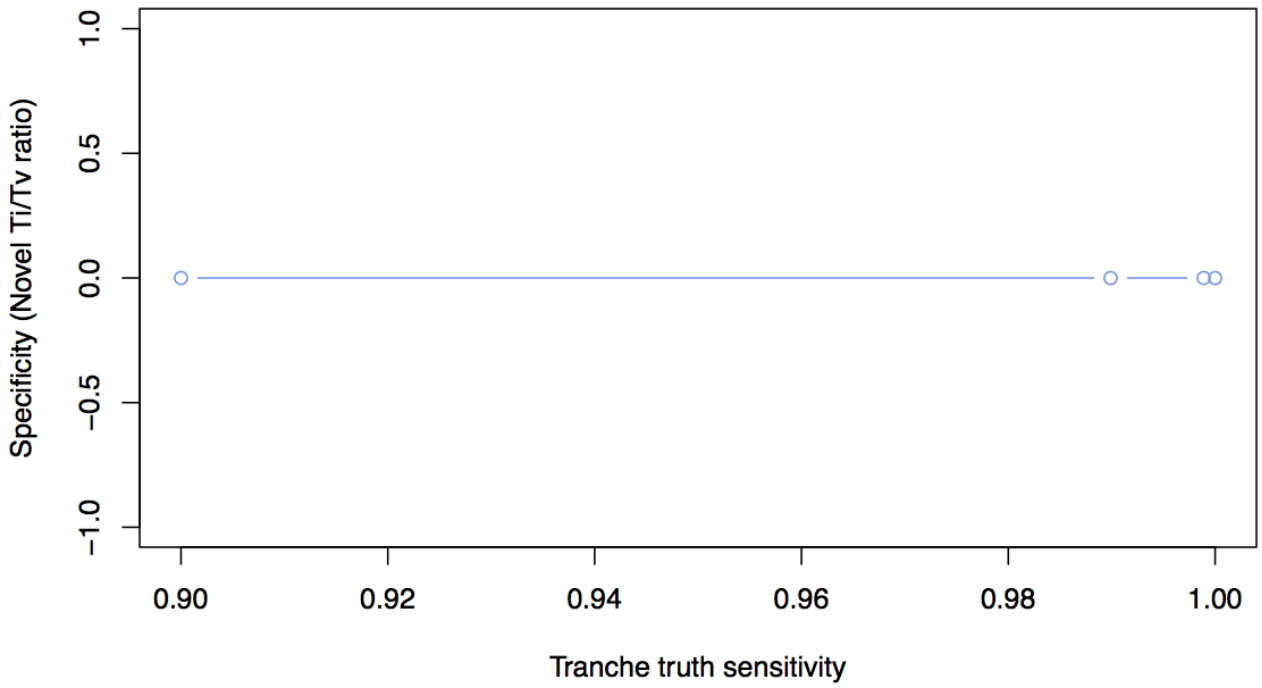
2.5.4 Build the Indel recalibration model

```
java -Xmx6g -jar $GATK/GenomeAnalysisTK.jar \
    -T VariantRecalibrator \
    -R ${reference} \
    -input ${result}/recalibrated_snps_raw_indels.vcf \
    -resource:mills,known=true,training=true,truth=true,prior=12.0 ${mills} \
    -an DP \
    -an FS \
    -an MQRankSum \
```

```
    -an ReadPosRankSum \
    -mode INDEL \
    -tranche 100.0 \
    -tranche 99.9 \
    -tranche 99.0 \
    -tranche 90.0 \
    -numBad 1000 \
    --maxGaussians 4 \
    -recalFile ${result}/recalibrate_INDEL.recal \
    -tranchesFile ${result}/recalibrate_INDEL.tranches \
    -rscriptFile ${result}/recalibrate_INDEL_plots.R
```

Again, a number of plots are generated by R

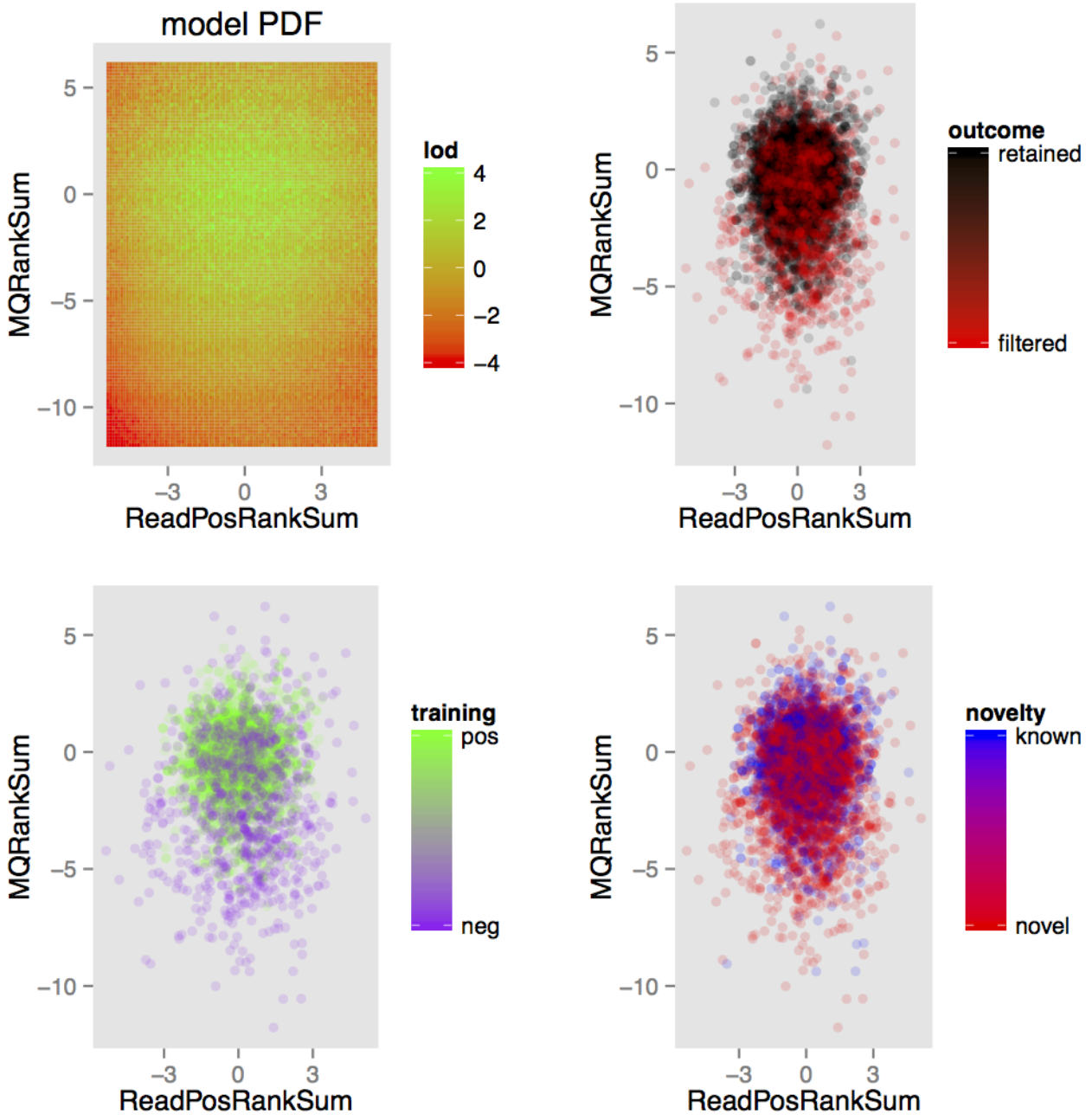




2.5.5 Apply the desired level of recalibration to the Indels in the call set

```
java -Xmx6g -jar $GATK/GenomeAnalysisTK.jar \
    -T ApplyRecalibration \
    -R ${reference} \
    -input ${result}/recalibrated_snps_raw_indels.vcf \
    -mode INDEL \
    --ts_filter_level 99.0 \
    -recalFile ${result}/recalibrate_INDEL.recal \
    -tranchesFile ${result}/recalibrate_INDEL.tranches \
    -o ${result}/recalibrated_variants.vcf
```

Detailed plots are provided for each step of the recalibration. We only reproduce here page #1 of 10 similar panels



The content of the file can be estimated based on the FILTER column

```
grep -v "^#" recalibrated_variants.vcf | cut -f 7 | sort | uniq -c
#    940 LowQual
#  77239 PASS
#    365 VQSRTrancheINDEL99.00to99.90
#    836 VQSRTrancheINDEL99.90to100.00
#   4291 VQSRTrancheSNP99.00to99.90
#   6667 VQSRTrancheSNP99.90to100.00
```

2.5.6 Filter high quality data for comparison analysis

The double-recalibrated file contains a FILTER field (column #7) that can be used to extract a high-quality subset

3 filter high quality variants using snpEff

```
# equivalent to filter "( ! FILTER='LowQual' )"
cat ${result}/recalibrated_variants.vcf | \
    java -Xmx6g -jar /opt/biotools/snpEff/SnpSift.jar \
    filter "( na FILTER ) | (FILTER = 'PASS' | FILTER =~ 'VQSRT')" \
    | bgzip -c > ${result}/hq_recalibrated_variants.vcf.gz && \
    tabix -p vcf ${result}/hq_recalibrated_variants.vcf.gz
```

3.0.7 Perform QC on the VCF calls

Two tools can be used that are part of VCFTools and HTSLIB

The older command **vcf-stats** will generate text files that can be consulted for a given metric.

```
vcf-stats recalibrated_variants.vcf.gz -p recalibrated_variants_stats/
vcf-stats hq_recalibrated_variants.vcf.gz -p hq_recalibrated_variants_stats/
```

The current version of HTSLib produces nicer output already seen in the BITS training.

```
htscmd vcfcheck recalibrated_variants.vcf.gz > recalibrated_variants.chk
htscmd vcfcheck hq_recalibrated_variants.vcf.gz > hq_recalibrated_variants.chk

plot-vcfcheck recalibrated_variants.chk -p recalibrated_variants_plots/
plot-vcfcheck hq_recalibrated_variants.chk -p hq_recalibrated_variants_plots/
```

For all GATK calls

	SNPs		indels			MNPs	others
Callset	n	ts/tv	n	frm*	3n**		
recal	74,796	2.01	15,625	–	0.50	0	0

* frameshift ratio: out/(out+in); ** 3n/non-3n

	singletons (AC=1)			multiallelic	
Callset	SNPs	ts/tv	indels	sites	SNPs
recal	69.3%	2.02	70.0%	829	68

- recal .. `recalibrated_variants.vcf.gz`

For the High Quality subset

	SNPs		indels			MNPs	others
Callset	n	ts/tv	n	frm*	3n**		
hq_re	74,383	2.01	15,098	–	0.47	0	0
			* frameshift ratio: out/(out+in);			** 3n/non-3n	

	singletons (AC=1)			multiallelic	
Callset	SNPs	ts/tv	indels	sites	SNPs
hq_re	69.2%	2.02	69.3%	829	68

- hq_re .. `hq_recalibrated_variants.vcf.gz`

3.1 Comparing GATK and Samtools calls with gold standard NA15807 variant lists

3.1.1 intersect results with former Training results

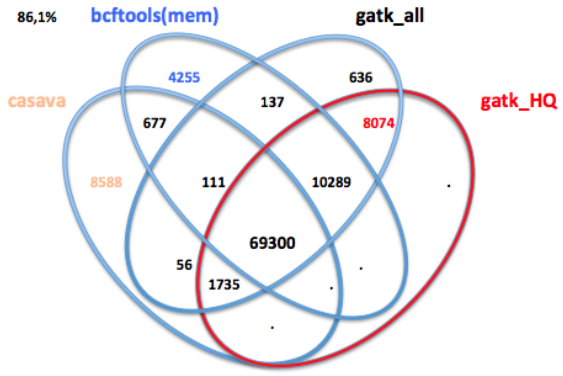
```
casava=public_info/Illumina_BaseSpace/chr21_NA18507_CASAVA-1.8_hg19.vcf.gz
bcftools=${base}/samtools_variants/chr21_mem_var.flt-D1000.vcf.gz
hapmap=public_info/NA18507_hapmap_3.3.hg19/chr21_NA18507_hapmap_3.3.hg19.vcf.gz
gatk_raw=${result}/recalibrated_variants.vcf.gz
gatk_hq=${result}/hq_recalibrated_variants.vcf.gz

vcf-compare ${casava} \
    ${bcftools} \
    ${gatk_raw} \
    ${gatk_hq} | grep "^VN" > ${result}/compare-4_2gatk.cmp
```

4-way venn comparison

		sum_all	%all
A	casava	80467	77,5%
B	bcftools(mem)	84769	81,6%
C	gatk_all	90338	87,0%
D	gatk_HQ	89398	86,1%

	region	count	pc1	pc2	pc3	pc4
VN	A	8588	A (10.7%)			
VN	AB	677	A (0.8%)	B (0.8%)		
VN	ABC	111	A (0.1%)	B (0.1%)	C (0.1%)	
VN	ABCD	69300	A (86.1%)	B (81.8%)	C (76.7%)	D (77.5%)
VN	AC	56	A (0.1%)		C (0.1%)	
VN	ACD	1735	A (2.2%)	C (1.9%)		D (1.9%)
VN	B	4255		B (5.0%)		
VN	BC	137		B (0.2%)	C (0.2%)	
VN	BCD	10289		B (12.1%)	C (11.4%)	D (11.5%)
VN	C	636			C (0.7%)	
VN	CD	8074			C (8.9%)	D (9.0%)
	absent	.				
	total	103858				

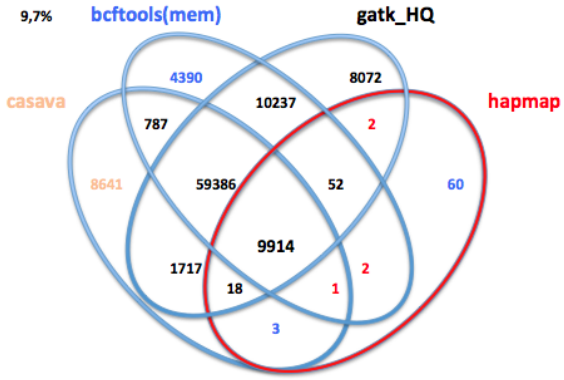



```
vcf-compare ${casava} \
    ${bcftools} \
    ${gatk_hq} \
    ${hapmap} | grep "^VN" > ${result}/compare-4_2hapmap.cmp
```

4-way venn comparison

		sum_all	%all
A	casava	80467	77,9%
B	bcftools(mem	84767	82,1%
C	gatk_HQ	81324	78,7%
D	hapmap	9990	9,7%

	region	count	pc1	pc2	pc3	pc4
VN	A	8641	A (10.7%)			
VN	AB	787	A (1.0%)	B (0.9%)		
VN	ABC	59386	A (73.8%)	B (70.1%)	C (66.4%)	
VN	ABCD	9914	A (12.3%)	B (11.7%)	C (11.1%)	D (98.6%)
VN	ABD	1	A (0.0%)	B (0.0%)		D (0.0%)
VN	AC	1717	A (2.1%)		C (1.9%)	
VN	ACD	18	A (0.0%)		C (0.0%)	D (0.2%)
VN	AD	3	A (0.0%)			D (0.0%)
VN	B	4390		B (5.2%)		
VN	BC	10237		B (12.1%)	C (11.5%)	
VN	BCD	52		B (0.1%)	C (0.1%)	D (0.5%)
VN	BD	2	B (0.0%)			D (0.0%)
VN	C	8072			C (9.0%)	
VN	CD	2	C (0.0%)			D (0.0%)
VN	D	60				D (0.6%)
	total	103282				



4 Discussion and conclusion

The full processing of the BWA mem mapping data for NA18507 and its comparison with the results available from reference sources and from the BITS mapping and calling experiment are very similar. A few % of the calls are private to one caller or to the public data. The comparison with the HapMap data (re validated information) shows a similar sensitivity for both methods.

The time necessary to perform this full workflow is of about 1/2 day while the corresponding part using **samtools** and **bcftools** runs in a few minutes. This time difference will be even more pronounced when dealing with a full genome analysis.

By contrast, GATK being accepted as the cleanest way to call high quality variant might motivate spending this extra time on the final stages of NGS DNA variant analysis

Another possibility, applied by many groups would be to take the intersection of both methods as a highest possible prediction quality in the case of an unknown genome.

In conclusion, we show here that applying GATK to call variants is a tedious but doable process. It requires time and computer power but can be run on a reasonable computer (strong laptop).

What was not explored here is the possibility to split this task in parallel jobs when doing full genome analysis of when analyzing multiple genomes. The GATK toolkit is provided with an additional toolbox called **QUEUE** that was not tried here but is meant to accelerate routine or heavy load analyses.

5 Appendices

6 wget-bundle.sh

```
#! /usr/bin/env bash
## script: 'wget-bundle.sh'
## ©SP-BITS, 2013 v1.0

# get all proper bundle files for version 2.5 at once
# ~375GB in total, this will take SOME time!!!

ver=2.5
local=$BIODATA/bundle_${ver}
mkdir -p ${local}

# loop in distant path and mirror all
for folder in exampleFASTA hg18 hg19 b36 b37; do

        echo "downloading data for "${folder}
        mkdir -p ${local}/${folder}
        cd ${local}/${folder}
        wget --ftp-user=gsapubftp-anonymous \
                --ftp-password="" \
                --retr-symlinks \
                -S -nd -np \
                --reject ".*,*.md5*,*.out" \
                -r ftp.broadinstitute.org:/bundle/${ver}/${folder}

        echo "# done for ${folder}"
done

# also get Liftover_Chain_Files
lcf=Liftover_Chain_Files
mkdir -p ${local}/${lcf}
cd ${local}/${lcf}

echo "downloading data for "${lcf}
wget --ftp-user=gsapubftp-anonymous \
                --ftp-password="" \
                --retr-symlinks \
                -S -nd -np \
                --reject ".*,*.md5*,*.out" \
                -r ftp.broadinstitute.org:/${lcf}

echo "# done for ${lcf}"
```

Contents