1st example

1st example

For this example, we extracted a 1000 base-region of a whole genome paired end sequencing experiment of a tumor and its matching control sample on the Illumina platform. This region contains a couple of variants, one of which appears to be a somatic variant only present in the tumor sample. We will use this example to go through some of the typical steps in a variant calling analysis, we will visualize the intermediate output files in a genome browser and we will take a detailed look at the file formats.

Files from Variant_calling/1_full alignments:

- Open the Integrative Genomics Viewer

```
java -jar igv.jar
```

- Load the two separate full bam files
- Select the correct genome to be displayed: Human (b37)
- Select the region covered by the bam file: chr9:131456000-131457000
- Explore the zooming and navigating features
- Inspect the variant position: chr9:131456174 ; in which gene falls this position? Which bases are read in the control and tumor samples, respectively?
- Leave the IGV open with the two bam files loaded
- Inform yourself on the options of samtools view:

```
samtools view
```

- Inspect the bam files and their header section running samtools view:

```
samtools view -h control.bam | less
```

- According to the bam file header, which is the sort order of the alignments and which program has been used for alignment?

Files from Variant_calling/2_with_readgroups:

- Load the two separate full bam files with readgroups in the IGV (in addition to the full ones from the previous step)
- Which differences do you see when mouse-over the reads from the first vs the read group step? Which additional information is extracted from the read group-containing bam files?
- Close the two bam files from the first step and their coverage tracks in the IGV; keep the two bam files with read groups open.
- Inspect the bam files and their header section running samtools view, as in the previous step. Which lines have been added to the header? Which fields have been added to the read records?

Files from Variant_calling/3_after_rmdup:

- Load the two separate bam files after rmdup in the IGV (in addition to the full ones with read groups from the previous step)
- Can you see a difference?
- Inspect the bam files and their header section. What information has been added to the header in the rmdup step?
- Run samtools view on the full bam files from step 1 or 2 and count the lines, then run samtools view on the bam files after rmdup and count the lines. How many reads have been removed in the control and in the tumor bam file in the rmdup step, respectively?

```
samtools view control.bam | wc -l
```

- Close the two bam files with read groups and their coverage tracks in the IGV; keep the two bam files after rmdup open.
- Inform yourself on the options of the GATK core engine and the realignment tests:

```
java -jar GenomeAnalysisTK.jar -h
java -jar GenomeAnalysisTK.jar -T RealignerTargetCreator -h
java -jar GenomeAnalysisTK.jar -T IndelRealigner -h
```

- Use the bam files after rmdup as input to run the local realignment in two steps:
    - Run GATK RealignerTargetCreator:

```
java -jar GenomeAnalysisTK.jar -T RealignerTargetCreator -I tumor_rmdup.bam -I
control_rmdup.bam -R human37.fa --known 1000G_phase1.indels.hg19.vcf --known
Mills_and_1000G_gold_standard.indels.hg19.vcf -o IndelRealigner.intervals -L chr9
```

- - Run GATK IndelRealigner:

```
java -jar GenomeAnalysisTK.jar -T IndelRealigner -I tumor_rmdup.bam -I
control_rmdup.bam -R human37.fa -targetIntervals IndelRealigner.intervals -L chr9 -o
samplepair.bam
```

Files from Variant_calling/4_realigned:

- Use the realigned bam file you generated yourself or the one provided in the folder.
- Inspect the bam file and its header section. What information has been added to the header in the realignment step?
- Run samtools view on the samplepair bam file and count the lines; compare to the numbers you counted for the two bam files after rmdup.
- Load the samplepair bam file in the IGV.
- In the IGV, group alignments by read group.
- Inspect the variant position again: chr9:131456174

Files from Variant_calling/5_pileup:

- The files in this folder were generated running samtools mpileup on the whole genomic region covered by the bam files. Use them to control your own results. We will use the control.RG and tumor.RG files for excluding specific read groups in one of he exercises, though.
- Inform yourself on the options of samtools mpileup:

```
samtools mpileup
```

- For practise, use the samplepair bam file as input and run samtools mpileup for the genomic interval shown in the presentation:

```
samtools mpileup -f human37.fa -r chr9:131456174-131456185 samplepair.bam
```

- What is the total coverage at our variant position of interest at chr9:131456174 ?
- Run samtools mpileup for the same interval, but exclude the tumor or the control sample, respectively:

```
samtools mpileup -f human37.fa -r chr9:131456174-131456185 -R tumor.RG samplepair.bam
samtools mpileup -f human37.fa -r chr9:131456174-131456185 -R control.RG
samplepair.bam
```

- What is the coverage at our variant position of interest at chr9:131456174 in the control and tumor sample, respectively, using these commands?
- What is the coverage at our variant position of interest at chr9:131456174 in the control and tumor sample, respectively, setting the base quality cutoff to 20? And setting the base quality cutoff to 30?

```
samtools mpileup -f human37.fa -r chr9:131456174-131456185 -Q 20 -R tumor.RG
samplepair.bam
samtools mpileup -f human37.fa -r chr9:131456174-131456185 -Q 20 -R control.RG
samplepair.bam
```

- Run samtools mpileup on the whole range of the bam file and browse through the output. Do you find any other possible variant positions, either in both samples or in one of them, apart from chr9:131456174 ?

```
samtools mpileup -f human37.fa samplepair.bam
samtools mpileup -f human37.fa -R tumor.RG samplepair.bam
samtools mpileup -f human37.fa -R control.RG samplepair.bam
```

Files from Variant_calling/6_vcf_samtools:

- Inform yourself on the options of bcftools view:

```
bcftools view
```

- Generate the vcf file in this folder yourself running samtools mpileup piped to bcftools view:

```
samtools mpileup -DSug -f human37.fa samplepair.bam | bcftools view -vcg - >
variants.vcf
```

- Look inside the vcf file: How many variants does your vcf file contain? How many of them are SNPs, how many are INDELs?
- Which is the most trustworthy position considering per-sample genotype qualities?
- Which is the most trustworthy position considering the per-sample genotype likelihoods?
- Which is the most trustworthy position considering all-over call quality?

Files from Variant_calling/7_vcf_GATK:

- Inform yourself on the options of the GATK UnifiedGenotyper test:

```
java -jar GenomeAnalysisTK.jar -T UnifiedGenotyper -h
```

- Generate the vcf file in this folder yourself running GATK UnifiedGenotyper:

```
java -jar GenomeAnalysisTK.jar -R human37.fa -T UnifiedGenotyper -I samplepair.bam -o
variants_GATK.vcf -L chr9:131456000-131457000 -glm BOTH -dt NONE
```

- Look inside the vcf file: How many variants does your vcf file contain? How many of them are SNPs, how many are INDELs?
- Compare the positions in this vcf file with the ones in the vcf file generated by samtools. Which positions are in both files, which are only in one of them?
- Look at the FORMAT field: Which values are output by samtools, and which by GATK? Look at those output by both programs, are they identical/similar/different?

- Open the vcf file as another track in the IGV along with the samplepair bam file and inspect how the variant positions are displayed there.

Files from Variant_calling/8_vcf_annotated_snpEff:

- Inform yourself on the options of the snpEff annotations command:

```
java -jar SNPEFF/snpEff.jar eff
```

- Generate the annotated vcf file in this folder yourself running snpEff with the samtools vcf file as input:

```
java -jar SNPEFF/snpEff.jar eff -c SNPEFF/snpEff.config -noLog -o vcf GRCh37.65
variants.vcf > annotated_variants.vcf
```

- Look inside the vcf file: How many different transcripts are annotated at each of the variant positions?
- How many variants overlap coding regions? Do they have an effect on amino acid encoding?
- What is the highest functional effect (HIGH, MODERATE, LOW or MODIFIER) annotated for each variant position?
- Have a look at the additional files that are created by the snpEff command: snpEff_summary.html and snpEff_genes.txt

Files from Variant_calling/9_vcf_annotated_dbSNP:

- We are not generating this file during this tutorial; please use the prepared file. Note that it has been generated using a different version of samtools and that the control and tumor files are named N402 and N401, respectively.
- Look inside the file: How many positions are annotated with dbSNP ID's (RS-numbers)?
- Look up the RS-numbers at http://www.ncbi.nlm.nih.gov/SNP

Files from Variant_calling/10_vcf_filtered:

- We are not generating this file during this tutorial; please use the prepared file. Note that it has been generated using a different version of samtools and that the control and tumor files are named N402 and N401, respectively.
- Look inside the file: Which tags have been added to each position? Read the description for each filter tag in the header section of the file and reproduce from the INFO and FORMAT section of the positions, why the tags were added.
- The position that has been tagged as somatic, overlaps a gene. Which gene is it? Look up more information on this gene in the following ressources: http://omim.org/ , http://genome.ucsc.edu/cgi-bin/hgTracks , http://www.ncbi.nlm.nih.gov/gap/phegeni