Introduction to DNA-Seq processing for cancer data

By Mathieu Bourgey, Ph.D

In this workshop, we will present the main steps that are commonly used to process and to analyze cancer sequencing data. We will focus only on whole genome data and provide command lines that allow detecting Single Nucleotide Variants (SNV). This workshop will show you how to launch individual steps of a complete DNA-Seq SNV pipeline using cancer data

Data Source

We will be working on a CageKid sample pair, patient C0098. The CageKid project is part of ICGC and is focused on renal cancer in many of it's forms. The raw data can be found on EGA and calls, RNA and DNA, can be found on the ICGC portal. For more details about CageKid

For practical reasons we subsampled the reads from the sample because running the whole dataset would take way too much time and resources.

Original Setup

The initial structure of your folders should look like this:

```
<ROOT>
|-- raw_reads/               # fastqs from the center (down sampled)
    `-- normal               # The blood sample directory
        `-- run*_?           # Lane directory by run number. Contains the fastqs
    `-- tumor                # The tumor sample directory
        `-- run*_?           # Lane directory by run number. Contains the fastqs
`-- project.nanuq.csv        # sample sheet
```

Cheat file

- You can find all the unix command lines of this practical in the file: commands.sh

Environment setup

```
export APP_ROOT=/home/training/Applications/
export PATH=$PATH:$APP_ROOT/IGVTools
export PICARD_JAR=$APP_ROOT/picard-tools/picard.jar
export SNPEFF_HOME=$APP_ROOT/snpEff/
export GATK_JAR=$APP_ROOT/gatk/GenomeAnalysisTK.jar
export BVATOOLS_JAR=$APP_ROOT/bvatools-1.6/bvatools-1.6-full.jar
export TRIMMOMATIC_JAR=$APP_ROOT/Trimmomatic-0.33/trimmomatic-0.33.jar
export STRELKA_HOME=$APP_ROOT/strelka-1.0.14/
export MUTECT_JAR=$APP_ROOT/mutect-src/mutect-1.1.7.jar
export VARSCAN_JAR=$APP_ROOT/varscan2/VarScan.v2.3.9.jar
export REF=/home/training/ebicancerworkshop201507/reference

cd $HOME/ebicancerworkshop201507
```

Software requirements

These are all already installed, but here are the original links.

- FastQC
- BVATools
- SAMTools
- IGV
- BWA
- Genome Analysis Toolkit
- Picard
- SnpEff
- MuTect
- Strelka

First data glance

So you've just received an email saying that your data is ready for download from the sequencing center of your choice.

What should you do ? *solution_data*

Fastq files

Let's first explore the fastq file.

Try these commands

```
zless -S raw_reads/normal/run62DVGAAXX_1/normal.64.pair1.fastq.gz
```

Why was it like that ? *solution_fastq1*

Now try these commands:

```
zcat raw_reads/normal/run62DVGAAXX_1/normal.64.pair1.fastq.gz | head -n4
zcat raw_reads/normal/run62DVGAAXX_1/normal.64.pair2.fastq.gz | head -n4
```

What was special about the output ?

Why was it like that? *solution_fastq2*

You could also just count the reads

```
zgrep -c "^@HWUSI" raw_reads/normal/run62DVGAAXX_1/normal.64.pair1.fastq.gz
```

We should obtain 4003 reads

Why shouldn't you just do ?

```
zgrep -c "^@" raw_reads/normal/run62DVGAAXX_1/normal.64.pair1.fastq.gz
```

solution_fastq3

Quality

We can't look at all the reads. Especially when working with whole genome 50x data. You could easily have Billions of reads.

Tools like FastQC and BVATools readsqc can be used to plot many metrics from these data sets.

Let's look at the data:

```
# Generate original QC
mkdir originalQC/
java -Xmx1G -jar ${BVATOOLS_JAR} readsqc --quality 64 \
  --read1 raw_reads/normal/run62DVGAAXX_1/normal.64.pair1.fastq.gz \
  --read2 raw_reads/normal/run62DVGAAXX_1/normal.64.pair2.fastq.gz \
  --threads 2 --regionName normalrun62DVGAAXX_1 --output originalQC/
```

Open the images

All the generated graphics have their uses. But 3 of them are particularly useful to get an overal picture of how good or bad a run went. - The Quality box plots

$$Q_{\text{sanger}} = -10 \log_{10} p$$

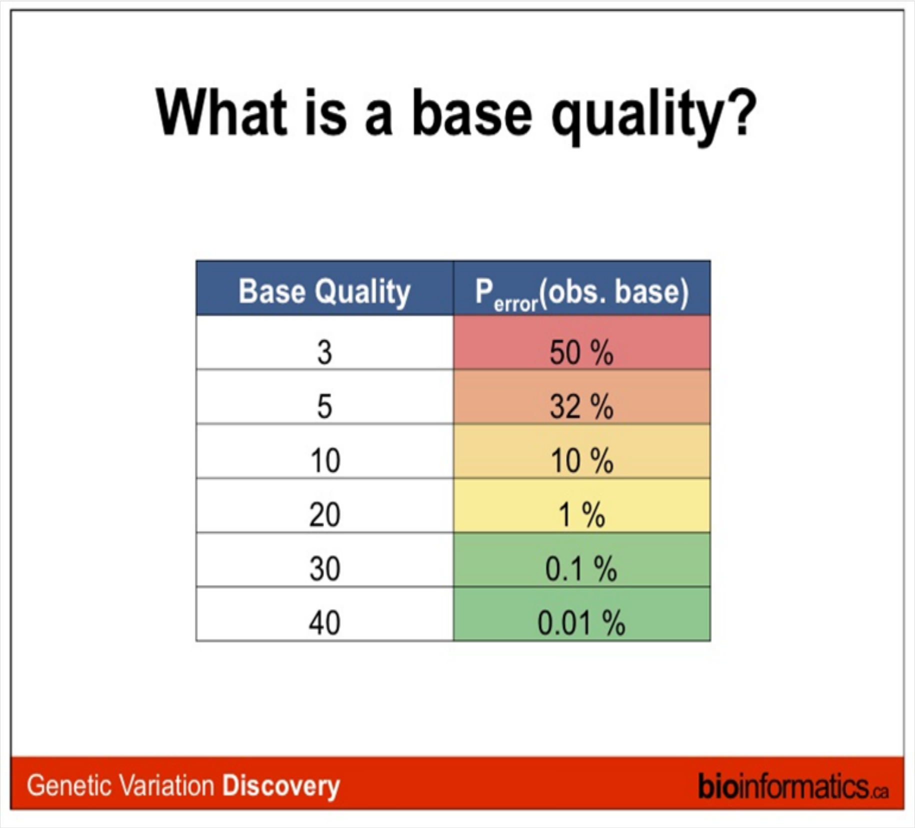
Figure 1: Phred quality score formula

- The nucleotide content graphs. - The Box plot shows the quality distribution of your data.

The quality of a base is computated using the Phread quality score. notes

The quality of a base is computated using the Phread quality score.

In the case of base quality the probability use represents the probability of base to have been wrongly called



What is a base quality?

Base Quality	P_{error}(obs. base)
3	50 %
5	32 %
10	10 %
20	1 %
30	0.1 %
40	0.01 %

Genetic Variation **Discovery**

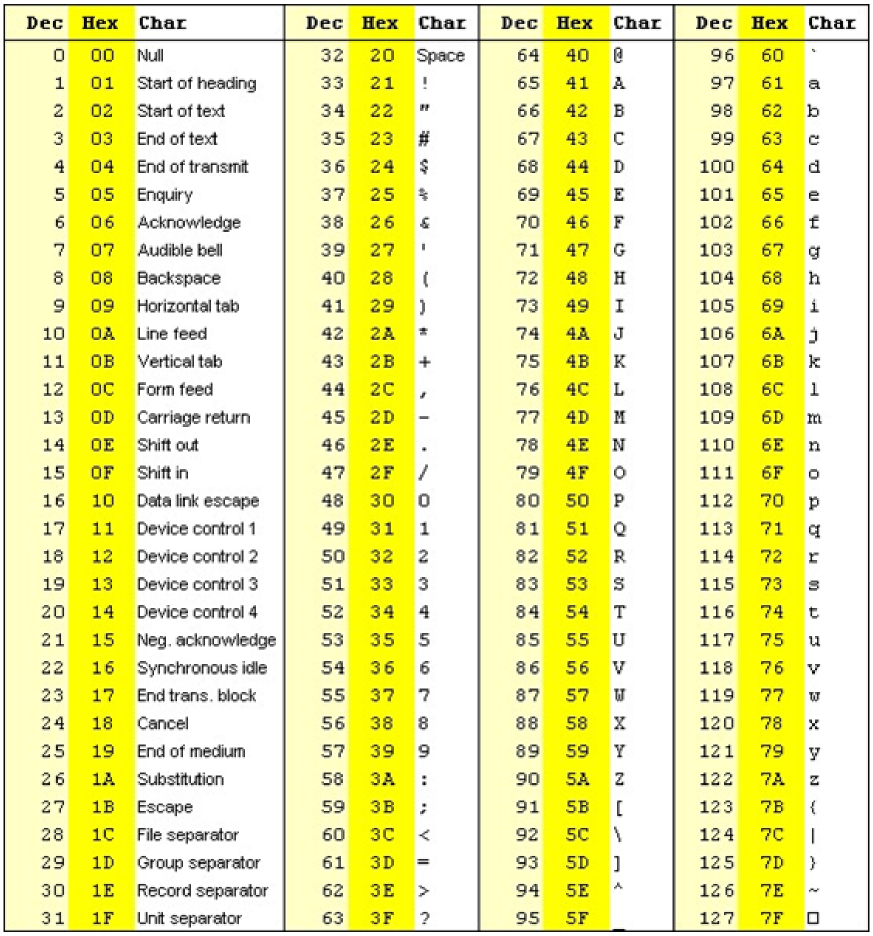
bioinformatics.ca

Figure 2: Base Quality values

The formula outputs an integer that is encoded using an ASCII table.

The way the lookup is done is by taking the the phred score adding 33 and using this number as a lookup in the table.

Older illumina runs, and the data here, were using phred+64 instead of phred+33 to encode their fastq files.



Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00	Null	32	20	Space	64	40	@	96	60	`
1	01	Start of heading	33	21	!	65	41	A	97	61	a
2	02	Start of text	34	22	"	66	42	B	98	62	b
3	03	End of text	35	23	#	67	43	C	99	63	c
4	04	End of transmit	36	24	$	68	44	D	100	64	d
5	05	Enquiry	37	25	%	69	45	E	101	65	e
6	06	Acknowledge	38	26	&	70	46	F	102	66	f
7	07	Audible bell	39	27	'	71	47	G	103	67	g
8	08	Backspace	40	28	(	72	48	H	104	68	h
9	09	Horizontal tab	41	29	)	73	49	I	105	69	i
10	0A	Line feed	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage return	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	47	2F	/	79	4F	O	111	6F	o
16	10	Data link escape	48	30	0	80	50	P	112	70	p
17	11	Device control 1	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	50	32	2	82	52	R	114	72	r
19	13	Device control 3	51	33	3	83	53	S	115	73	s
20	14	Device control 4	52	34	4	84	54	T	116	74	t
21	15	Neg. acknowledge	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	54	36	6	86	56	V	118	76	v
23	17	End trans. block	55	37	7	87	57	W	119	77	w
24	18	Cancel	56	38	8	88	58	X	120	78	x
25	19	End of medium	57	39	9	89	59	Y	121	79	y
26	1A	Substitution	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	60	3C	<	92	5C	\	124	7C	|
29	1D	Group separator	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	63	3F	?	95	5F	_	127	7F	□

Figure 3: ACII table

What stands out in the graphs ? *solution_fastqQC1*

Why do we see adapters ? *solution_adapter1*

Although nowadays this doesn't happen often, it does still happen. In some cases, miRNA, it is expected to have adapters.

Trimming

Since adapter are not part of the genome they should be removed

To do that we will use Trimmomatic.

The adapter file is in your work folder.

```
cat adapters.fa
```

Why are there 2 different ones ? *solution_trim1*

trimming with trimmomatic:

```
# Trim and convert data
for file in raw_reads/*/run*_?/*.pair1.fastq.gz;
do
  FNAME=`basename $file`;
  DIR=`dirname $file`;
  OUTPUT_DIR=`echo $DIR | sed 's/raw_reads/reads/g'`;

  mkdir -p $OUTPUT_DIR;
  java -Xmx2G -cp $TRIMMOMATIC_JAR org.usadellab.trimmomatic.TrimmomaticPE -threads 2 -phred
    $file \
    ${file%.pair1.fastq.gz}.pair2.fastq.gz \
    ${OUTPUT_DIR}/${FNAME%.64.pair1.fastq.gz}.t30l50.pair1.fastq.gz \
    ${OUTPUT_DIR}/${FNAME%.64.pair1.fastq.gz}.t30l50.single1.fastq.gz \
    ${OUTPUT_DIR}/${FNAME%.64.pair1.fastq.gz}.t30l50.pair2.fastq.gz \
    ${OUTPUT_DIR}/${FNAME%.64.pair1.fastq.gz}.t30l50.single2.fastq.gz \
    TOPHRED33 ILLUMINACLIP:adapters.fa:2:30:15 TRAILING:30 MINLEN:50 \
    2> ${OUTPUT_DIR}/${FNAME%.64.pair1.fastq.gz}.trim.out ;
done

cat reads/normal/run62DVGAAXX_1/normal.trim.out
```

note on trimmomatic command

What does Trimmomatic says it did ? *solution_trim2*

Exercice: **Let's generate the new graphs** *solution_fastqQC2*

How does it look now ? *solution_trim3*

Alignment

The raw reads are now cleaned up of artefacts we can align each lane separatly.

Why should this be done separatly? *solution_aln1*

Why is it important to set Read Group information ? [Solution](solutions_aln2*

Alignment with bwa-mem

```
# Align data
for file in reads/*/run62*_4/*.pair1.fastq.gz;
do
  FNAME=`basename $file`;
  DIR=`dirname $file`;
  OUTPUT_DIR=`echo $DIR | sed 's/reads/alignment/g'`;
  SNAME=`echo $file | sed 's/reads\/\([^/]\+\)\/.*/\1/g'`;
  RUNID=`echo $file | sed 's/.*\/run\([^_]\+\)_.*/\1/g'`;
  LANE=`echo $file | sed 's/.*\/run[^_]\+_\(.\).*/\1/g'`;

  mkdir -p $OUTPUT_DIR;

  bwa mem -M -t 3 \
    -R "@RG\\tID:${SNAME}_${RUNID}_${LANE}\\tSM:${SNAME}\\t\
LB:${SNAME}\\tPU:${RUNID}_${LANE}\\tCN:Centre National de Genotypage\\tPL:ILLUMINA" \
    ${REF}/Homo_sapiens.GRCh37.fa \
    $file \
    ${file%.pair1.fastq.gz}.pair2.fastq.gz \
  | java -Xmx2G -jar ${PICARD_JAR}  SortSam \
    INPUT=/dev/stdin \
    OUTPUT=${OUTPUT_DIR}/${SNAME}.sorted.bam \
    CREATE_INDEX=true VALIDATION_STRINGENCY=SILENT SORT_ORDER=coordinate MAX_RECORDS_IN_RAM=
done
```

Why did we pipe the output of one to the other ? *solution_aln3*

Could we have done it differently ? *solution_aln4*

Lane merging

We now have alignments for each of the sequences lanes:

- This is not practical in it's current form.
- What we wan't to do now is merge the results into one BAM.

Since we identified the reads in the BAM with read groups, even after the merging, we can still identify the origin of each read.

```
# Merge Data
java -Xmx2G -jar ${PICARD_JAR}  MergeSamFiles \
  INPUT=alignment/normal/run62DPDAAXX_8/normal.sorted.bam \
  INPUT=alignment/normal/run62DVGAAXX_1/normal.sorted.bam \
  INPUT=alignment/normal/run62MK3AAXX_5/normal.sorted.bam \
  INPUT=alignment/normal/runA81DF6ABXX_1/normal.sorted.bam \
  INPUT=alignment/normal/runA81DF6ABXX_2/normal.sorted.bam \
  INPUT=alignment/normal/runBC04D4ACXX_2/normal.sorted.bam \
  INPUT=alignment/normal/runBC04D4ACXX_3/normal.sorted.bam \
  INPUT=alignment/normal/runBD06UFACXX_4/normal.sorted.bam \
  INPUT=alignment/normal/runBD06UFACXX_5/normal.sorted.bam \
  OUTPUT=alignment/normal/normal.sorted.bam \
  VALIDATION_STRINGENCY=SILENT CREATE_INDEX=true

java -Xmx2G -jar ${PICARD_JAR}  MergeSamFiles \
  INPUT=alignment/tumor/run62DU0AAXX_8/tumor.sorted.bam \
  INPUT=alignment/tumor/run62DUUAAXX_8/tumor.sorted.bam \
  INPUT=alignment/tumor/run62DVMAAXX_4/tumor.sorted.bam \
  INPUT=alignment/tumor/run62DVMAAXX_6/tumor.sorted.bam \
  INPUT=alignment/tumor/run62DVMAAXX_8/tumor.sorted.bam \
  INPUT=alignment/tumor/run62JREAAXX_4/tumor.sorted.bam \
  INPUT=alignment/tumor/run62JREAAXX_6/tumor.sorted.bam \
  INPUT=alignment/tumor/run62JREAAXX_8/tumor.sorted.bam \
  INPUT=alignment/tumor/runAC0756ACXX_5/tumor.sorted.bam \
  INPUT=alignment/tumor/runBD08K8ACXX_1/tumor.sorted.bam \
  INPUT=alignment/tumor/run62DU6AAXX_8/tumor.sorted.bam \
  INPUT=alignment/tumor/run62DUYAAXX_7/tumor.sorted.bam \
  INPUT=alignment/tumor/run62DVMAAXX_5/tumor.sorted.bam \
  INPUT=alignment/tumor/run62DVMAAXX_7/tumor.sorted.bam \
  INPUT=alignment/tumor/run62JREAAXX_3/tumor.sorted.bam \
  INPUT=alignment/tumor/run62JREAAXX_5/tumor.sorted.bam \
  INPUT=alignment/tumor/run62JREAAXX_7/tumor.sorted.bam \
  INPUT=alignment/tumor/runAC0756ACXX_4/tumor.sorted.bam \
  INPUT=alignment/tumor/runAD08C1ACXX_1/tumor.sorted.bam \
  OUTPUT=alignment/tumor/tumor.sorted.bam \
  VALIDATION_STRINGENCY=SILENT CREATE_INDEX=true
```

You should now have one BAM containing all your data.

Let's double check

```
ls -l alignment/normal/
samtools view -H alignment/normal/normal.sorted.bam | grep "^@RG"
```

You should have your 9 read group entries.

Why did we use the -H switch? *solution_merge1*

Try without. What happens? *solution_merge2*

lane merging note

SAM/BAM exploration

Let's spend some time to explore bam files.

```
samtools view alignment/normal/normal.sorted.bam | head -n4
```

Here you have examples of alignment results. A full description of the flags can be found in the SAM specification http://samtools.sourceforge.net/SAM1.pdf

You can try using picards explain flag site to understand what is going on with your reads http://broadinstitute.github.io/picard/explain-flags.html

The flag is the 2nd column.

What do the flags of the first 4th reads mean? [solutions](solutions/_sambam1*

Exercice: **Let's take the 3rd one, the one that is in proper pair, and find it's mate.** [solutions](solutions/_sambam3*

Why the pairing information is important ? [solutions](solutions/_sambam4*

SAM/BAM filtering

You can use samtools to filter reads as well.

Exercice: **How many reads mapped and unmapped were there?** *solution_sambam2*

SAM/BAM CIGAR string

Another useful bit of information in the SAM is the CIGAR string. It's the 6th column in the file.

This column explains how the alignment was achieved.

```
    M == base aligns *but doesn't have to be a match*. A SNP will have an M even if it disag
    I == Insertion
    D == Deletion
    S == soft-clips. These are handy to find un removed adapters, viral insertions, etc.
```

An in depth explanation of the CIGAR can be found here

The exact details of the cigar string can be found in the SAM spec as well.

We won't go into too much detail at this point since we want to concentrate on cancer specific issues now.

Cleaning up alignments

We started by cleaning up the raw reads. Now we need to fix some alignments.

The first step for this is to realign around indels and snp dense regions.

The Genome Analysis toolkit has a tool for this called IndelRealigner.

It basically runs in 2 steps:

1. Find the targets
2. Realign them

GATK IndelRealigner

```
# Realign
java -Xmx2G  -jar ${GATK_JAR} \
  -T RealignerTargetCreator \
  -R ${REF}/Homo_sapiens.GRCh37.fa \
  -o alignment/normal/realign.intervals \
  -I alignment/normal/normal.sorted.bam \
  -I alignment/tumor/tumor.sorted.bam \
  -L 9

java -Xmx2G -jar ${GATK_JAR} \
  -T IndelRealigner \
  -R ${REF}/Homo_sapiens.GRCh37.fa \
  -targetIntervals alignment/normal/realign.intervals \
  --nWayOut .realigned.bam \
  -I alignment/normal/normal.sorted.bam \
  -I alignment/tumor/tumor.sorted.bam

  mv normal.sorted.realigned.ba* alignment/normal/
  mv tumor.sorted.realigned.ba* alignment/tumor/
```

Why did we use both normal and tumor together? *solution_realign3*

How could we make this go faster ? *solution_realign1*

How many regions did it think needed cleaning ? *solution_realign2*

Indel Realigner also makes sure the called deletions are left aligned when there is a microsatellite or homopolymer.

```
This
ATCGAAAA-TCG
into
```

```
ATCG-AAAATCG

or
ATCGATATATATA--TCG
into
ATCG--ATATATATATCG
```

Why it is important ?*solution_realign4*

FixMates

Why ?

- Some read entries don't have their mate information written properly.

We use Picard to do this:

```
# Fix Mate
java -Xmx2G -jar ${PICARD_JAR}  FixMateInformation \
  VALIDATION_STRINGENCY=SILENT CREATE_INDEX=true SORT_ORDER=coordinate MAX_RECORDS_IN_RAM=5
  INPUT=alignment/normal/normal.sorted.realigned.bam \
  OUTPUT=alignment/normal/normal.matefixed.bam
java -Xmx2G -jar ${PICARD_JAR}  FixMateInformation \
  VALIDATION_STRINGENCY=SILENT CREATE_INDEX=true SORT_ORDER=coordinate MAX_RECORDS_IN_RAM=5
  INPUT=alignment/tumor/tumor.sorted.realigned.bam \
  OUTPUT=alignment/tumor/tumor.matefixed.bam
```

Mark duplicates

What are duplicate reads ? *solution_markdup1*

What are they caused by ? *solution_markdup2*

What are the ways to detect them ? *solution_markdup3*

Here we will use picards approach:

```
# Mark Duplicates
java -Xmx2G -jar ${PICARD_JAR}  MarkDuplicates \
  REMOVE_DUPLICATES=false CREATE_MD5_FILE=true VALIDATION_STRINGENCY=SILENT CREATE_INDEX=tru
  INPUT=alignment/normal/normal.matefixed.bam \
  OUTPUT=alignment/normal/normal.sorted.dup.bam \
  METRICS_FILE=alignment/normal/normal.sorted.dup.metrics
```

```
java -Xmx2G -jar ${PICARD_JAR}  MarkDuplicates \
  REMOVE_DUPLICATES=false CREATE_MD5_FILE=true VALIDATION_STRINGENCY=SILENT CREATE_INDEX=tru
  INPUT=alignment/tumor/tumor.matefixed.bam \
  OUTPUT=alignment/tumor/tumor.sorted.dup.bam \
  METRICS_FILE=alignment/tumor/tumor.sorted.dup.metrics
```

We can look in the metrics output to see what happened.

```
less alignment/normal/normal.sorted.dup.metrics
```

How many duplicates were there ? *solution_markdup4*

We can see that it computed separate measures for each library.

Why is this important to do not combine everything ? *solution_markdup5*

Note on Duplicate rate

Base Quality recalibration

Why do we need to recalibrate base quality scores ? *solution_recal1*

It runs in 2 steps, 1- Build covariates based on context and known snp sites 2- Correct the reads based on these metrics

GATK BaseRecalibrator:

```
# Recalibrate
for i in normal tumor
do
  java -Xmx2G -jar ${GATK_JAR} \
    -T BaseRecalibrator \
    -nct 2 \
    -R ${REF}/Homo_sapiens.GRCh37.fa \
    -knownSites ${REF}/dbSnp-137_chr9.vcf.gz \
    -L 9:130215000-130636000 \
    -o alignment/${i}/${i}.sorted.dup.recalibration_report.grp \
    -I alignment/${i}/${i}.sorted.dup.bam

    java -Xmx2G -jar ${GATK_JAR} \
      -T PrintReads \
      -nct 2 \
      -R ${REF}/Homo_sapiens.GRCh37.fa \
      -BQSR alignment/${i}/${i}.sorted.dup.recalibration_report.grp \
      -o alignment/${i}/${i}.sorted.dup.recal.bam \
      -I alignment/${i}/${i}.sorted.dup.bam
done
```

“‘

Just to see how things change let’s make GATK recalibrate after a first pass

“‘{.bash}

Extract BAM metrics

Once your whole bam is generated, it’s always a good thing to check the data again to see if everything makes sens.

Compute coverage If you have data from a capture kit, you should see how well your targets worked

Insert Size It tells you if your library worked

Alignment metrics It tells you if your sample and you reference fit together

Compute coverage

Both GATK and BVATools have depth of coverage tools.

Here we’ll use the GATK one

```
# Get Depth
for i in normal tumor
do
  java  -Xmx2G -jar ${GATK_JAR} \
    -T DepthOfCoverage \
    --omitDepthOutputAtEachBase \
    --summaryCoverageThreshold 10 \
    --summaryCoverageThreshold 25 \
    --summaryCoverageThreshold 50 \
    --summaryCoverageThreshold 100 \
    --start 1 --stop 500 --nBins 499 -dt NONE \
    -R ${REF}/Homo_sapiens.GRCh37.fa \
    -o alignment/${i}/${i}.sorted.dup.recal.coverage \
    -I alignment/${i}/${i}.sorted.dup.recal.bam \
    -L 9:130215000-130636000
done
```

note on DepthOfCoverage command

Coverage is the expected ~70-110x in these project

Look at the coverage:

```
less -S alignment/normal/normal.sorted.dup.recal.coverage.sample_interval_summary
less -S alignment/tumor/tumor.sorted.dup.recal.coverage.sample_interval_summary
```

Is the coverage fit with the expectation ? *solution_DOC1*

Insert Size

It corresponds to the size of DNA fragments sequenced.

Different from the gap size (= distance between reads) !

These metrics are computed using Picard:

```
# Get insert size
for i in normal tumor
do
  java -Xmx2G -jar ${PICARD_JAR}  CollectInsertSizeMetrics \
    VALIDATION_STRINGENCY=SILENT \
    REFERENCE_SEQUENCE=${REF}/Homo_sapiens.GRCh37.fa \
    INPUT=alignment/${i}/${i}.sorted.dup.recal.bam \
    OUTPUT=alignment/${i}/${i}.sorted.dup.recal.metric.insertSize.tsv \
    HISTOGRAM_FILE=alignment/${i}/${i}.sorted.dup.recal.metric.insertSize.histo.pdf \
    METRIC_ACCUMULATION_LEVEL=LIBRARY
done
```

look at the output

```
less -S alignment/normal/normal.sorted.dup.recal.metric.insertSize.tsv
less -S alignment/tumor/tumor.sorted.dup.recal.metric.insertSize.tsv
```

There is something interesting going on with our library ERR.

Can you tell what it is? *solution_insert1*

Alignment metrics

For the alignment metrics, samtools flagstat is very fast but with bwa-mem since some reads get broken into pieces, the numbers are a bit confusing.

We prefer the Picard way of computing metrics:

```
# Get alignment metrics
for i in normal tumor
do
```

```
  java -Xmx2G -jar ${PICARD_JAR}  CollectAlignmentSummaryMetrics \
    VALIDATION_STRINGENCY=SILENT \
    REFERENCE_SEQUENCE=${REF}/Homo_sapiens.GRCh37.fa \
    INPUT=alignment/${i}/${i}.sorted.dup.recal.bam \
    OUTPUT=alignment/${i}/${i}.sorted.dup.recal.metric.alignment.tsv \
    METRIC_ACCUMULATION_LEVEL=LIBRARY
done
```

explore the results

```
less -S alignment/normal/normal.sorted.dup.recal.metric.alignment.tsv
less -S alignment/tumor/tumor.sorted.dup.recal.metric.alignment.tsv
```

Do you think the sample and the reference genome fit together ? *solution_alnMetrics1*

Variant calling

Most of SNV caller use either a Baysian, a threshold or a t-test approach to do the calling

Here we will try 3 variant callers. - Varscan 2 - MuTecT - Strelka

Other candidates - Virmid - Somatic sniper

many, MANY others can be found here: https://www.biostars.org/p/19104/

In our case, let's start with:

```
mkdir pairedVariants
```

varscan 2

VarScan calls somatic variants (SNPs and indels) using a heuristic method and a statistical test based on the number of aligned reads supporting each allele.

Varscan somatic caller expects both a normal and a tumor file in SAMtools pileup format. from sequence alignments in binary alignment/map (BAM) format. To build a pileup file, you will need:

- A SAM/BAM file ("myData.bam") that has been sorted using the sort command of SAMtools.
- The reference sequence ("reference.fasta") to which reads were aligned, in FASTA format.

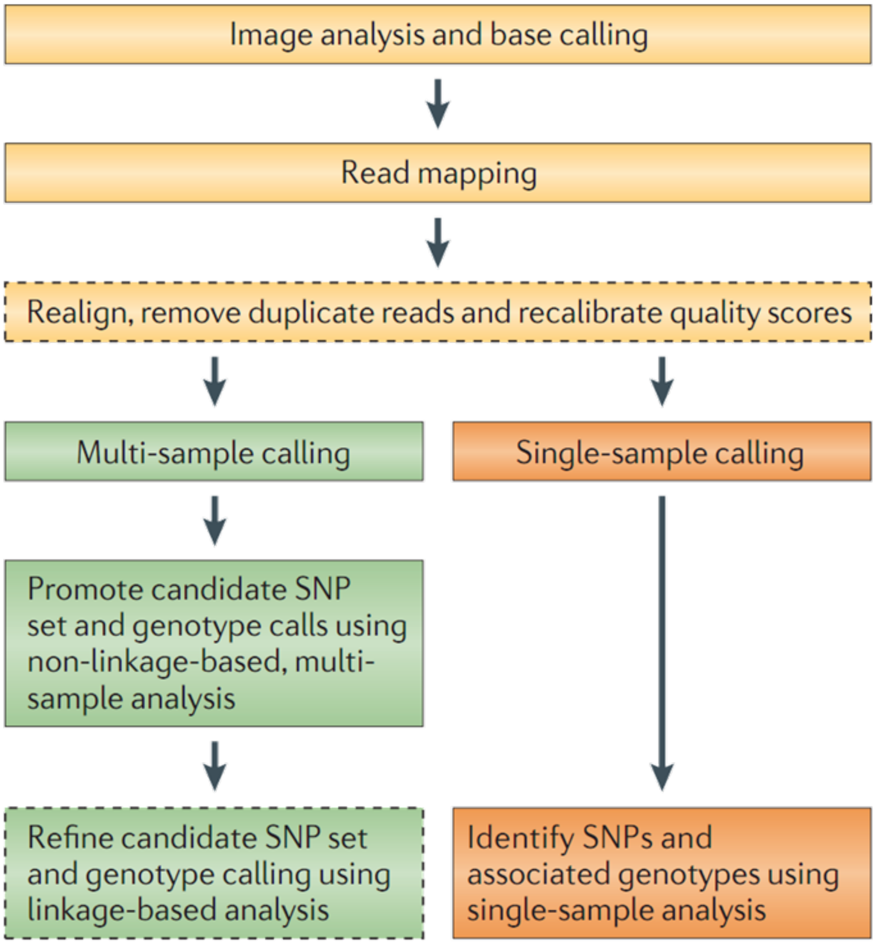


Image analysis and base calling
Read mapping
Realign, remove duplicate reads and recalibrate quality scores
Multi-sample calling
Single-sample calling
Promote candidate SNP set and genotype calls using non-linkage-based, multi-sample analysis
Refine candidate SNP set and genotype calling using linkage-based analysis
Identify SNPs and associated genotypes using single-sample analysis

Figure 4: SNV call summary workflow

- The SAMtools software package.

```
# SAMTools mpileup
for i in normal tumor
do
samtools mpileup -L 1000 -B -q 1 \
  -f ${REF}/Homo_sapiens.GRCh37.fa \
  -r 9:130215000-130636000 \
  alignment/${i}/${i}.sorted.dup.recal.bam \
  > pairedVariants/${i}.mpileup
done

# varscan
java -Xmx2G -jar ${VARSCAN_JAR} somatic pairedVariants/normal.mpileup pairedVariants/tumor.m
```

note on samtools mpileup and bcftools command

Broad MuTecT

```
# Variants MuTecT
# Note MuTecT only works with Java 6, 7 will give you an error
# if you get "Comparison method violates its general contract!
# you used java 7"
java -Xmx2G -jar ${MUTECT_JAR} \
  -T MuTect \
  -R ${REF}/Homo_sapiens.GRCh37.fa \
  -dt NONE -baq OFF --validation_strictness LENIENT -nt 2 \
  --dbsnp ${REF}/dbSnp-137_chr9.vcf.gz \
  --cosmic ${REF}/b37_cosmic_v70_140903.vcf.gz \
  --input_file:normal alignment/normal/normal.sorted.dup.recal.bam \
  --input_file:tumor alignment/tumor/tumor.sorted.dup.recal.bam \
  --out pairedVariants/mutect.call_stats.txt \
  --coverage_file pairedVariants/mutect.wig.txt \
  -pow pairedVariants/mutect.power \
  -vcf pairedVariants/mutect.vcf \
  -L 9:130215000-130636000
```

Illumina Strelka

```
# Variants Strelka
cp ${STRELKA_HOME}/etc/strelka_config_bwa_default.ini ./
# Fix ini since we subsampled
sed 's/isSkipDepthFilters =.*/isSkipDepthFilters = 1/g' -i strelka_config_bwa_default.ini
```

```
${STRELKA_HOME}/bin/configureStrelkaWorkflow.pl \
  --normal=alignment/normal/normal.sorted.dup.recal.bam \
  --tumor=alignment/tumor/tumor.sorted.dup.recal.bam \
  --ref=${REF}/Homo_sapiens.GRCh37.fa \
  --config=$(pwd)/strelka_config_bwa_default.ini \
  --output-dir=pairedVariants/strelka/

  cd pairedVariants/strelka/
  make -j3
  cd ../..

  cp pairedVariants/strelka/results/passed.somatic.snvs.vcf pairedVariants/strelka.vcf
```

Now we have variants from all three methods. Let's compress and index the vcfs for futur visualisation.

```
for i in pairedVariants/*.vcf;do bgzip -c $i > $i.gz ; tabix -p vcf $i.gz;done
```

Let's look at a compressed vcf.

```
zless -S pairedVariants/varscan.snp.vcf.gz
```

Details on the spec can be found here: http://vcftools.sourceforge.net/specs.html

Fields vary from caller to caller.

Some values are are almost always there:

- The ref vs alt alleles,
- variant quality (QUAL column)
- The per-sample genotype (GT) values.

note on the vcf format fields

Annotations

We typically use snpEff but many use annovar and VEP as well.

Let's run snpEff:

```
# SnpEff
java  -Xmx6G -jar ${SNPEFF_HOME}/snpEff.jar \
  eff -v -c ${SNPEFF_HOME}/snpEff.config \
  -o vcf \
  -i vcf \
  -stats pairedVariants/varscan.snpeff.vcf.stats.html \
  hg19 \
  pairedVariants/paired_varscan.snp.vcf \
  > pairedVariants/varscan.snpeff.vcf
```

Look at the new vcf file:

```
less -S pairedVariants/mpileup.snpeff.vcf
```

Can you see the difference with the previous vcf ? *solution_snpeff1*

Exercice: **Find a somatic mutation with a predicted High or Moderate impact** *solution_snpeff2*

For now we will not explore this step since you will be working with gene annotations in your next workshop.

You could also take a look at the HTML stats file snpEff created: it contains some metrics on the variants it analyzed.

Data visualisation

The Integrative Genomics Viewer (IGV) is an efficient visualization tool for interactive exploration of large genome datasets.

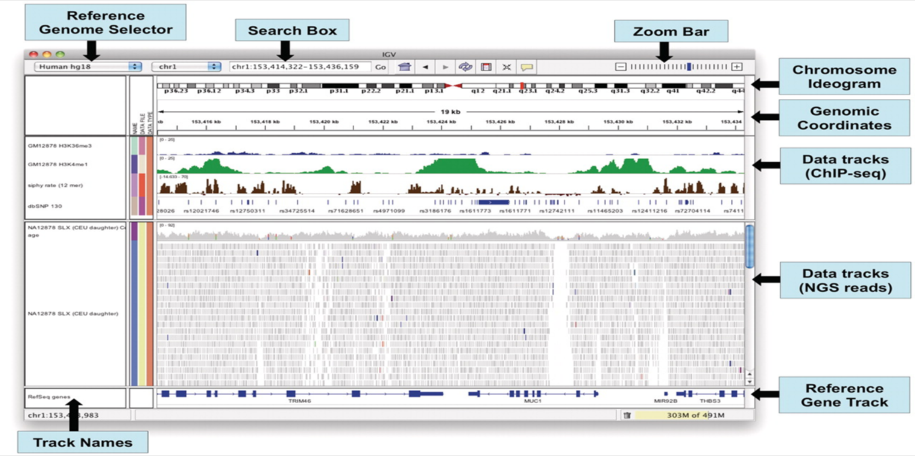


Figure 5: IGV browser presentation

Before jumping into IGV, we'll generate a track IGV can use to plot coverage:

```
# Coverage Track
for i in normal tumor
do
  igvtools count \
    -f min,max,mean \
    alignment/${i}/${i}.sorted.dup.recal.bam \
    alignment/${i}/${i}.sorted.dup.recal.bam.tdf \
    b37
done
```

Then:

1. Open IGV
2. Chose the reference genome corresponding to those use for alignment (b37)
3. Load bam file
4. Load vcf files

Explore/play with the data:

- find somatic variants
- Look around…

solution_igv1

Aknowledgments

This tutorial is an adaptation of the one created by Louis letourneau here. I would like to thank and acknowledge Louis for this help and for sharing his material. The format of the tutorial has been inspired from Mar Gonzalez Porta. I also want to acknowledge Joel Fillon, Louis Letrouneau (again), Francois Lefebvre, Maxime Caron and Guillaume Bourque for the help in building these pipelines and working with all the various datasets.