Cancer Genomics SNV, Exploratory practical - Cancer Genomics Workshop 2014

Louis Letourneau

2014/07/02

Contents

Introduction to DNA-Seq processing

This workshop will show you how to launch individual steps of a complete DNA-Seq pipeline for Cancer analysis

Source Material

All the documentation for the practicals can be found on github here:
https://github.com/lletourn/Workshops/tree/ebiCancerWorkshop201407

Data Soruce

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: http://www.cng.fr/cagekid/

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

This work is licensed under a Creative Commons Attribution-ShareAlike 3.0 Unported License[1]. This means that you are able to copy, share and modify the work, as long as the result is distributed under the same license.

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 sheets

- Unix comand line cheat sheet[2]

Environment setup

```
export APP_ROOT=/home/training/Applications/
export PATH=$PATH:$APP_ROOT/bwa-0.7.9a:$APP_ROOT/tabix-0.2.6/:$APP_ROOT/IGVTools
export PICARD_HOME=$APP_ROOT/picard-tools-1.115/
export SNPEFF_HOME=$APP_ROOT/snpEff/
export GATK_JAR=$APP_ROOT/gatk/GenomeAnalysisTK.jar
export BVATOOLS_JAR=$APP_ROOT/bvatools-1.3/bvatools-1.3-full.jar
export TRIMMOMATIC_JAR=$APP_ROOT/Trimmomatic-0.32/trimmomatic-0.32.jar
export STRELKA_HOME=$APP_ROOT/strelka-1.0.13/
export REF=/home/training/ebiCancerWorkshop201407/references/

cd $HOME/ebiCancerWorkshop201407
```

Software requirements

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

[1]http://creativecommons.org/licenses/by-sa/3.0/deed.en_US
[2]http://sites.tufts.edu/cbi/files/2013/01/linux_cheat_sheet.pdf

- FastQC[3]
- BVATools[4]
- SAMTools[5]
- IGV[6]
- BWA[7]
- Genome Analysis Toolkit[8]
- Picard[9]
- SnpEff[10]
- MuTect[11]
- Strelka[12]

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. The first thing to do is download it, the second thing is making sure it is of good quality.

Fastq files

Let's first explore the fastq file.

Try these commands

```
zless -S raw_reads/normal/runD0YR4ACXX_1/normal.64.pair1.fastq.gz

zcat raw_reads/normal/runD0YR4ACXX_1/normal.64.pair1.fastq.gz | head -n4
zcat raw_reads/normal/runD0YR4ACXX_1/normal.64.pair2.fastq.gz | head -n4
```

From the second set of commands (the head), what was special about the output? Why was it like that? Solution[13]

You could also just count the reads

```
zgrep -c "^@HISEQ2" raw_reads/normal/runD0YR4ACXX_1/normal.64.pair1.fastq.gz
```

Why shouldn't you just do

```
zgrep -c "^@" raw_reads/normal/runD0YR4ACXX_1/normal.64.pair1.fastq.gz
```

Solution[14]

[3] http://www.bioinformatics.babraham.ac.uk/projects/fastqc/
[4] https://bitbucket.org/mugqic/bvatools/downloads
[5] http://sourceforge.net/projects/samtools/
[6] http://www.broadinstitute.org/software/igv/download
[7] http://bio-bwa.sourceforge.net/
[8] http://www.broadinstitute.org/gatk/
[9] http://picard.sourceforge.net/
[10] http://snpeff.sourceforge.net/
[11] http://www.broadinstitute.org/cancer/cga/mutect
[12] https://sites.google.com/site/strelkasomaticvariantcaller/
[13] https://github.com/lletourn/Workshops/blob/ebiCancerWorkshop201407/solutions/_fastq.ex1.md
[14] https://github.com/lletourn/Workshops/blob/ebiCancerWorkshop201407/solutions/_fastq.ex2.md

Quality

We can't look at all the reads. Especially when working with whole genome 50x data. You could easilly 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/
java7 -Xmx1G -jar ${BVATOOLS_JAR} readsqc --quality 64 \
  --read1 raw_reads/normal/runD0YR4ACXX_1/normal.64.pair1.fastq.gz \
  --read2 raw_reads/normal/runD0YR4ACXX_1/normal.64.pair2.fastq.gz \
  --threads 2 --regionName normalD0YR4ACXX_1 --output originalQC/
```

Open the images.

What stands out in the graphs? Solution[15]

All the generated graphics have their uses. This being said 2 of them are particularly useful to get an overal picture of how good or bad a run went. These are the Quality box plots and the nucleotide content graphs.

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

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

The formula outputs an integer that is encoded using an ASCII[16] 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. The Wikipedia entry for the FASTQ format[17] has a summary of the varying values.

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

We see a little bit of adapter in the sequences. Why does this happen Solution[18]

Trimming

After this careful analysis of the raw data we see that:

- Some reads have bad 3' ends.
- Some reads have adapter sequences in them.
- Data needs to be converted into phred+33 from phred+64

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

Since they are not part of the genome of interest they should be removed if enough reads have them.

To be able to remove the adapters we need to feed them to a tool. In this case we will use Trimmomatic. The adapter file is already in your work folder. We can look at the adapters

[15]https://github.com/lletourn/Workshops/blob/ebiCancerWorkshop201407/solutions/_fastqQC.ex1.md
[16]http://en.wikipedia.org/wiki/ASCII
[17]http://en.wikipedia.org/wiki/FASTQ_format
[18]https://github.com/lletourn/Workshops/blob/ebiCancerWorkshop201407/solutions/_fastqQC.ex2.md

```
cat adapters.fa
```

Why are there 2 different ones? Solution[19]

Another reason we want to run Trimmomatic here is to convert the data from phred+33 to phred+64. Trimmomatic does this directly. In modern datasets this is not needed.

Let's try removing them and see what happens.

```
# 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;
  java7 -Xmx2G -cp $TRIMMOMATIC_JAR org.usadellab.trimmomatic.TrimmomaticPE -threads 2 -phred64 \
    $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/runD0YR4ACXX_1/normal.trim.out
```

What does Trimmomatic says it did? Solution[20]

Since the data was so good to start with, we won't regenerate the graph post trim (or you could do it as an extra exercise if you wish)

Alignment

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

Why should this be done separatly? Solution[21]

```
# Align data
for file in reads/*/run*_?/*.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'`;
```

[19]https://github.com/lletourn/Workshops/blob/ebiCancerWorkshop201407/solutions/_trim.ex1.md
[20]https://github.com/lletourn/Workshops/blob/ebiCancerWorkshop201407/solutions/_trim.ex2.md
[21]https://github.com/lletourn/Workshops/blob/ebiCancerWorkshop201407/solutions/_aln.ex1.md

```
  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}/bwa/b37.fasta \
    $file \
    ${file%.pair1.fastq.gz}.pair2.fastq.gz \
  | java7 -Xmx2G -jar ${PICARD_HOME}/SortSam.jar \
    INPUT=/dev/stdin \
    OUTPUT=${OUTPUT_DIR}/${SNAME}.sorted.bam \
    CREATE_INDEX=true VALIDATION_STRINGENCY=SILENT SORT_ORDER=coordinate MAX_RECORDS_IN_RAM=500000
done
```

Why is it important to set Read Group information? Solution[22]

The details of the fields can be found in the SAM/BAM specifications Here[23] For most cases, only the sample name, platform unit and library one are important.

Why did we pipe the output of one to the other? Could we have done it differently? Solution[24]

We will explore the generated BAM latter.

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
java7 -Xmx2G -jar ${PICARD_HOME}/MergeSamFiles.jar \
  INPUT=alignment/normal/runC0LWRACXX_1/normal.sorted.bam \
  INPUT=alignment/normal/runC0LWRACXX_6/normal.sorted.bam \
  INPUT=alignment/normal/runC0PTAACXX_6/normal.sorted.bam \
  INPUT=alignment/normal/runC0PTAACXX_7/normal.sorted.bam \
  INPUT=alignment/normal/runC0PTAACXX_8/normal.sorted.bam \
  INPUT=alignment/normal/runC0R2BACXX_6/normal.sorted.bam \
  INPUT=alignment/normal/runC0R2BACXX_7/normal.sorted.bam \
  INPUT=alignment/normal/runC0R2BACXX_8/normal.sorted.bam \
  INPUT=alignment/normal/runD0YR4ACXX_1/normal.sorted.bam \
  INPUT=alignment/normal/runD0YR4ACXX_2/normal.sorted.bam \
  OUTPUT=alignment/normal/normal.sorted.bam \
  VALIDATION_STRINGENCY=SILENT CREATE_INDEX=true
```

[22]https://github.com/lletourn/Workshops/blob/ebiCancerWorkshop201407/solutions/_aln.ex2.md
[23]http://samtools.sourceforge.net/SAM1.pdf
[24]https://github.com/lletourn/Workshops/blob/ebiCancerWorkshop201407/solutions/_aln.ex3.md

```
java7 -Xmx2G -jar ${PICARD_HOME}/MergeSamFiles.jar \
  INPUT=alignment/tumor/runBC0TV0ACXX_8/tumor.sorted.bam \
  INPUT=alignment/tumor/runC0LVJACXX_6/tumor.sorted.bam \
  INPUT=alignment/tumor/runC0PK4ACXX_7/tumor.sorted.bam \
  INPUT=alignment/tumor/runC0PK4ACXX_8/tumor.sorted.bam \
  INPUT=alignment/tumor/runC0R29ACXX_7/tumor.sorted.bam \
  INPUT=alignment/tumor/runC0R29ACXX_8/tumor.sorted.bam \
  INPUT=alignment/tumor/runC0TTBACXX_3/tumor.sorted.bam \
  INPUT=alignment/tumor/runD114WACXX_8/tumor.sorted.bam \
  OUTPUT=alignment/tumor/tumor.sorted.bam \
  VALIDATION_STRINGENCY=SILENT CREATE_INDEX=true
```

You should now have one normal and one tumor 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 10 read group entries. Why did we use the `-H` switch? Try without. What happens? Solution[25]

SAM/BAM

Let's spend some time to explore bam files.

```
samtools view alignment/normal/normal.sorted.bam | head -n2
```

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://picard.sourceforge.net/explain-flags.html

The flag is the 2nd column.

You can use samtools to filter.

```
# Say you want to count the *un-aligned* reads you can use
samtools view -c -f4 alignment/normal/normal.sorted.bam

# Or you want to count the *aligned* reads you can use
samtools view -c -F4 alignment/normal/normal.sorted.bam
```

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

How many reads mapped and unmapped were there? Solution[26]

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.

[25]https://github.com/lletourn/Workshops/blob/ebiCancerWorkshop201407/solutions/_merge.ex1.md
[26]https://github.com/lletourn/Workshops/blob/ebiCancerWorkshop201407/solutions/_sambam.ex3.md

- M == base aligns *but doesn't have to be a match.* A SNP will have an M even if it disagrees with the reference.
- 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 there:
http://genome.sph.umich.edu/wiki/SAM
The exact details of the cigar string can be found in the SAM spec as well.
http://samtools.github.io/hts-specs/SAMv1.pdf

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.

For cancer there is a subtility

```
# Realign
java7 -Xmx2G  -jar ${GATK_JAR} \
  -T RealignerTargetCreator \
  -R ${REF}/b37.fasta \
  -o alignment/normal/realign.intervals \
  -I alignment/normal/normal.sorted.bam \
  -I alignment/tumor/tumor.sorted.bam \
  -L 19

java7 -Xmx2G -jar ${GATK_JAR} \
  -T IndelRealigner \
  -R ${REF}/b37.fasta \
  -targetIntervals alignment/normal/realign.intervals \
  --nWayOut .realigned.bam \
  -I alignment/normal/normal.sorted.bam \
  -I alignment/tumor/tumor.sorted.bam

  mv normal.sorted.realigned.bam alignment/normal/
  mv tumor.sorted.realigned.bam alignment/tumor/
```

Why did we use both normal and tumor together? Solution[27]

How could we make this go faster? Solution[28]
How many regions did it think needed cleaning? Solution[29]

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

[27]https://github.com/lletourn/Workshops/blob/ebiCancerWorkshop201407/solutions/_realign.ex3.md
[28]https://github.com/lletourn/Workshops/blob/ebiCancerWorkshop201407/solutions/_realign.ex1.md
[29]https://github.com/lletourn/Workshops/blob/ebiCancerWorkshop201407/solutions/_realign.ex2.md

```
This
ATCGAAAA-TCG
into
ATCG-AAAATCG

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

This makes it easier for down stream tools.

FixMates

This step shouldn't be necessary... but it is.

This goes through the BAM file and find entries which don't have their mate information written properly.

This used to be a problem in the GATKs realigner, but they fixed it. It shouldn't be a problem with aligners like BWA, but there are always corner cases that create one-off coordinates and such.

This happened a lot with bwa backtrack. This happens less with bwa mem, but it still happens none the less.

```
# Fix Mate
java7 -Xmx2G -jar ${PICARD_HOME}/FixMateInformation.jar \
  VALIDATION_STRINGENCY=SILENT CREATE_INDEX=true SORT_ORDER=coordinate MAX_RECORDS_IN_RAM=500000 \
  INPUT=alignment/normal/normal.sorted.realigned.bam \
  OUTPUT=alignment/normal/normal.matefixed.bam
java7 -Xmx2G -jar ${PICARD_HOME}/FixMateInformation.jar \
  VALIDATION_STRINGENCY=SILENT CREATE_INDEX=true SORT_ORDER=coordinate MAX_RECORDS_IN_RAM=500000 \
  INPUT=alignment/tumor/tumor.sorted.realigned.bam \
  OUTPUT=alignment/tumor/tumor.matefixed.bam
```

Mark duplicates

As the step says, this is to mark duplicate reads. What are duplicate reads? What are they caused by? Solution[30]

What are the ways to detect them? Solution[31]

Here we will use picards approach:

```
# Mark Duplicates
java7 -Xmx2G -jar ${PICARD_HOME}/MarkDuplicates.jar \
  REMOVE_DUPLICATES=false CREATE_MD5_FILE=true VALIDATION_STRINGENCY=SILENT CREATE_INDEX=true \
  INPUT=alignment/normal/normal.matefixed.bam \
```

[30]https://github.com/lletourn/Workshops/blob/ebiCancerWorkshop201407/solutions/_markdup.ex1.md
[31]https://github.com/lletourn/Workshops/blob/ebiCancerWorkshop201407/solutions/_markdup.ex2.md

```
  OUTPUT=alignment/normal/normal.sorted.dup.bam \
  METRICS_FILE=alignment/normal/normal.sorted.dup.metrics

java7 -Xmx2G -jar ${PICARD_HOME}/MarkDuplicates.jar \
  REMOVE_DUPLICATES=false CREATE_MD5_FILE=true VALIDATION_STRINGENCY=SILENT CREATE_INDEX=true \
  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. We can see that it computed seperate measures for each library. Why is this important to do and not combine everything? Solution[32]

How many duplicates were there? Solution[33]

This is pretty spot on for a whole genome project. <5% should be expected.

Recalibration

This is the last BAM cleaning up step.

The goal for this step is to try to recalibrate base quality scores. The vendors tend to inflate the values of the bases in the reads. Also, this step tries to lower the scores of some biased motifs for some technologies.

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

```
# Recalibrate
for i in normal tumor
do
  java7 -Xmx2G -jar ${GATK_JAR} \
    -T BaseRecalibrator \
    -nct 2 \
    -R ${REF}/b37.fasta \
    -knownSites ${REF}/dbSnp-137.vcf.gz \
    -L 19:50500000-52502000 \
    -o alignment/${i}/${i}.sorted.dup.recalibration_report.grp \
    -I alignment/${i}/${i}.sorted.dup.bam

    java7 -Xmx2G -jar ${GATK_JAR} \
      -T PrintReads \
      -nct 2 \
      -R ${REF}/b37.fasta \
      -BQSR alignment/${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

[32]https://github.com/lletourn/Workshops/blob/ebiCancerWorkshop201407/solutions/_markdup.ex3.md
[33]https://github.com/lletourn/Workshops/blob/ebiCancerWorkshop201407/solutions/_markdup.ex4.md

```
# Check Recalibration
for i in normal tumor
do
  java7 -Xmx2G -jar ${GATK_JAR} \
    -T BaseRecalibrator \
    -nct 2 \
    -R ${REF}/b37.fasta \
    -knownSites ${REF}/dbSnp-137.vcf.gz \
    -L 19:50500000-52502000 \
    -o alignment/${i}/${i}.sorted.dup.recalibration_report.seconnd.grp \
    -I alignment/${i}/${i}.sorted.dup.bam \
    -BQSR alignment/${i}/${i}.sorted.dup.recalibration_report.grp

  java7 -Xmx2G -jar ${GATK_JAR} \
    -T AnalyzeCovariates \
    -R ${REF}/b37.fasta \
    -before alignment/${i}/${i}.sorted.dup.recalibration_report.grp \
    -after alignment/${i}/${i}.sorted.dup.recalibration_report.seconnd.grp \
    -csv alignment/${i}/BQSR.${i}.csv \
    -plots alignment/${i}/BQSR.${i}.pdf
done
```

The graphs don't mean much because we downsampled the data quite a bit. With a true whole genome or whole exome dataset we can see a bigger effect.

Extract 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

Both GATK and BVATools have depth of coverage tools. We wrote our own in BVAtools because - GATK was deprecating theirs, but they changed their mind - GATK's is very slow - We were missing some output that we wanted from the GATK's one (GC per interval, valid pairs, etc)

Here we'll use the GATK one

```
# Get Depth
for i in normal tumor
do
  java7  -Xmx2G -jar ${GATK_JAR} \
    -T DepthOfCoverage \
    --omitDepthOutputAtEachBase \
    --summaryCoverageThreshold 10 \
    --summaryCoverageThreshold 25 \
    --summaryCoverageThreshold 50 \
    --summaryCoverageThreshold 100 \
    --start 1 --stop 500 --nBins 499 -dt NONE \
    -R ${REF}/b37.fasta \
    -o alignment/${i}/${i}.sorted.dup.recal.coverage \
    -I alignment/${i}/${i}.sorted.dup.recal.bam \
    -L 19:50500000-52502000 &
done
wait

# 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
```

Coverage is the expected ~70-110x. summaryCoverageThreshold is a usefull function to see if your coverage is uniform. Another way is to compare the mean to the median. If both are almost equal, your coverage is pretty flat. If both are quite different That means something is wrong in your coverage. A mix of WGS and WES would show very different mean and median values.

Insert Size

Now we extract the insert size of the fragments. This can be informative to see how well the library was constructed

```
# Get insert size
for i in normal tumor
do
  java -Xmx2G -jar ${PICARD_HOME}/CollectInsertSizeMetrics.jar \
    VALIDATION_STRINGENCY=SILENT \
    REFERENCE_SEQUENCE=${REF}/b37.fasta \
    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
```

Alignment metrics

For the alignment metrics, we used to use `samtools flagstat` but with bwa mem since some reads get broken into pieces, the numbers are a bit confusing. You can try it if you want.

We prefer the Picard way of computing metrics

```
# Get alignment metrics
for i in normal tumor
do
  java -Xmx2G -jar ${PICARD_HOME}/CollectAlignmentSummaryMetrics.jar \
    VALIDATION_STRINGENCY=SILENT \
    REFERENCE_SEQUENCE=${REF}/b37.fasta \
    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
```

Variant calling

Here we will try 3 variant callers. - SAMtools - MuTecT - Strelka

Other candidates - Varscan 2 - 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
```

SAMtools

```
# Variants SAMTools
samtools mpileup -L 1000 -B -q 1 -D -S -g \
  -f ${REF}/b37.fasta \
  -r 19:50500000-52502000 \
  alignment/normal/normal.sorted.dup.recal.bam \
  alignment/tumor/tumor.sorted.dup.recal.bam \
  | bcftools view -vcg -T pair - \
  > pairedVariants/mpileup.vcf
```

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}/b37.fasta \
  -dt NONE -baq OFF --validation_strictness LENIENT -nt 2 \
  --dbsnp ${REF}/dbSnp-137.vcf \
  --cosmic ${REF}/b37_cosmic_v54_120711.vcf \
  --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 19:50500000-52502000
```

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}/b37.fasta \
  --config=strelka_config_bwa_default.ini \
  --output-dir=pairedVariants/strelka/

  cd pairedVariants/strelka/
  make -j3
  cd $HOME/ebiCancerWorkshop201407

  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 variants/mpileup.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 more constant. The ref vs alt alleles, variant quality (QUAL column) and the per-sample genotype (GT) values are almost always there.

Annotations

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

Let's run snpEff

```
# SnpEff
java7  -Xmx6G -jar ${SNPEFF_HOME}/snpEff.jar \
  eff -v -c ${SNPEFF_HOME}/snpEff.config \
  -o vcf \
  -i vcf \
  -stats pairedVariants/mpileup.snpeff.vcf.stats.html \
  hg19 \
  pairedVariants/mpileup.vcf \
  > pairedVariants/mpileup.snpeff.vcf

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

We can see in the vcf that snpEff added a few sections. These are hard to decipher directly from the VCF other tools or scripts, need to be used to make sens of this.

For now we will skip this step since you will be working with gene annotations in your next workshop.

Take a look at the HTML stats file snpEff created. It contains some metrics on the variants it analysed.

Visualisation

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
```

IGV

You can get IGV here[34]

Open it and choose b37 as the genome

Open your BAM file, the tdf we just generated should load. Load your vcfs as well.

Find an indel. What's different between the snp callers? Solution[35] Go to 19:50500000-52502000 what is interesting here? Solution[36]

Look around…

Aknowledgments

The format for this tutorial has been inspired from Mar Gonzalez Porta of Embl-EBI, who I would like to thank and acknowledge. I also want to acknowledge Mathieu Bourgey, Francois Lefebvre, Maxime Caron and Guillaume Bourque for the help in building these pipelines and working with all the various datasets.

[34]http://www.broadinstitute.org/software/igv/download
[35]https://github.com/lletourn/Workshops/blob/ebiCancerWorkshop201407/solutions/_vis.ex1.md
[36]https://github.com/lletourn/Workshops/blob/ebiCancerWorkshop201407/solutions/_vis.ex2.md

Exploratory analysis of cancer data

This workshop will be a mix of different methods to look and explore your data.

We will be working on the same BAMs you generated from the SNV part. Again, for practical reasons we subsampled the reads from the sample because running the whole dataset would take way too much time and resources. This leads to some strange results in this part.

This work is licensed under a Creative Commons Attribution-ShareAlike 3.0 Unported License[37]. This means that you are able to copy, share and modify the work, as long as the result is distributed under the same license.

Environment setup

We will need an updated bvatools for these exercises

```
cd $HOME/ebiCancerWorkshop201407
wget "https://bitbucket.org/mugqic/bvatools/downloads/bvatools-dev.jar"

export BVATOOLS_JAR=$HOME/ebiCancerWorkshop201407/bvatools-dev.jar
export APP_ROOT=/home/training/Applications/
export PATH=$PATH:$APP_ROOT/bedtools2/bin
export PICARD_HOME=$APP_ROOT/picard-tools-1.115/
export SNPEFF_HOME=$APP_ROOT/snpEff/
export GATK_JAR=$APP_ROOT/gatk/GenomeAnalysisTK.jar
export TRIMMOMATIC_JAR=$APP_ROOT/Trimmomatic-0.32/trimmomatic-0.32.jar
export STRELKA_HOME=$APP_ROOT/strelka-1.0.13/
export REF=/home/training/ebiCancerWorkshop201407/references/

cd $HOME/ebiCancerWorkshop201407
```

Software requirements

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

- Genome MuSiC[38]
- BVATools[39]
- SAMTools[40]
- IGV[41]
- Genome Analysis Toolkit[42]
- Picard[43]
- SnpEff[44]

[37]http://creativecommons.org/licenses/by-sa/3.0/deed.en_US
[38]http://gmt.genome.wustl.edu/genome-shipit/genome-music/current/
[39]https://bitbucket.org/mugqic/bvatools/downloads
[40]http://sourceforge.net/projects/samtools/
[41]http://www.broadinstitute.org/software/igv/download
[42]http://www.broadinstitute.org/gatk/
[43]http://picard.sourceforge.net/
[44]http://snpeff.sourceforge.net/

Exome AI

Calling CNVs in exome is not simple. We have a tool that can help. Follow the instructions from the ExomeAI pdf[45], we will try it out.

http://genomequebec.mcgill.ca/exomeai

Telomeres

In this first step we will try to qualitatively see if the normal and tumor have different telomere lengths.

One way to do this is find the telomere motif.

A good link to get various telomere repeats is the Telomerase Database[46]

First step, count the number of reads with these repeats.

```
# Aligned or not, we want them all
samtools view alignment/normal/normal.sorted.bam | awk '{if($10 ~ /TTAGGGTTAGGGTTAGGG/) {SUM++}} END
samtools view alignment/tumor/tumor.sorted.bam | awk '{if($10 ~ /TTAGGGTTAGGGTTAGGG/) {SUM++}} END {p
```

Why did we put multiple copied of the repeat in the search? Solution[47]

Next look in the alignments summary file we generated yesterday and extract the number of aligned reads.

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

Now the rest can be done in good old excel.

Open a sheet up, load up your numbers and compute the fold change between normal and tumor.

This case is rather boring, there practically is no change.

Significantly Mutated Genes

For this part we will try to find significantly mutated genes. To do this we will use Genome MuSiC

First off there are few things are needed to make MuSiC work

```
mkdir genomeMusic
cd genomeMusic
```

Then:

1. A region of interest file (ROI)

[45]https://github.com/lletourn/Workshops/blob/ebiCancerWorkshop201407/doc/ExomeAI.pdf
[46]http://telomerase.asu.edu/sequences_telomere.html
[47]https://github.com/lletourn/Workshops/blob/ebiCancerWorkshop201407/solutions/_telo.ex1.md

2. A Mutation Annotation Format file (MAF)
3. A bam list

For the ROI we could use UCSC or BioMart to extract these. We will use Ensembl biomart. Go to www.ensembl.org/biomart/martview/

- Choose Database Ensembl Genes 75
- Choose Homo Sapiens Genes
- In filters (click en the left heading), Limit to Genes with RefSeq mRNA ID(s), and choose only chromosome 19 to make the statistical test run faster.
- Under Attributes
  - Choose Structure. Now click in this order (to have a BED file tyoe output)
  - Chromosome Name
  - Exon Start
  - Exon End
  - Associated Gene Name

Download the mart_results.txt file put the file in your genomeMusic directory.

Removed the header line (the first line from the file) Then we need to convert the tab file to a bed.

```
awk '{OFS="\t"} {print $1,$2-3,$3+2,$4}' mart_export.txt | sort -k1V,1V -k2n,2n > exons.roi.bed
```

Why did we substract 3 from the start column? Solution[48]

Merge overlaps to not count twice.

```
bedtools merge -nms -i exons.roi.bed | sed 's/;.*//g' > exons.roi.merged.bed
```

The sed command is to remove GENE;GENE;GENE when bedtools merges. This is not ideal though when genes overlap. A more complicated way should be used

Now for the bam list

```
echo -e \
"Sample\t$HOME/ebiCancerWorkshop201407/alignment/normal/normal.sorted.dup.recal.bam\t$HOME/ebiCancerW
> bam_list.fofn
```

Let's compute the coverage to see what's usable between samples (we'll come back to this)

```
mkdir musicOutput
gmt music bmr calc-covg --roi-file exons.roi.merged.bed \
  --bam-list bam_list.fofn --output-dir musicOutput/ \
  --reference-sequence $REF/b37.fasta
```

Now let's generate the MAF file

[48]https://github.com/lletourn/Workshops/blob/ebiCancerWorkshop201407/solutions/_music.ex1.md

```
java7 -Xmx3G -jar $BVATOOLS_JAR vcf2maf \
  --vcf ../pairedVariants/mpileup.snpeff.vcf \
  --minQual 70 --minCLR 45 \
  --forceTumorName Sample \
  --output ../pairedVariants/mpileup.snpeff.maf

# Usually you'll have one maf per patient so you'll need to merge them
cat ../pairedVariants/*.maf > genomeMusic.maf
```

Now there's a problem here. This version of bvatools requires a mapper which we didn't generate. A new version will come out shortly that fixes this. In the mean time, modify the MAF

- Add SPIB to the first entry
- Add KLK1,KLKP1,ETFB,SIGLEC8 to the last 4

Now compute the background rate

```
gmt music bmr calc-bmr --roi-file exons.roi.merged.bed \
  --reference-sequence $REF/b37.fasta --bam-list bam_list.fofn \
  --output-dir musicOutput/ --maf-file genomeMusic.maf
```

And finally compute the significance list

```
gmt music smg --gene-mr-file musicOutput/gene_mrs --output-file musicOutput/smg
```

Now you should have 2 files smg smg_details

smg_details has the p-values computed for all genes in your ROI file. smg only contains the genes for which at least 2 out of the 3 statistical test have an FDR <= max-fdr which is 0.2 by default.

In this case with one subsampled sample, the results don't make much sens.

Substitution plots

After calling somatic mutations on WGS, we often want to see

- What is the transition counts vs transversion counts
- Where do these mutations fall, Intergenic, UTR5', CDS, etc

There are a milion ways to do this, we will try one.

First, as we did with MuSiC we need to figure out what parts of the genome are callable accross all the samples. To do this we will use a GATK tool called CallableLoci

```
for i in normal tumor
do
  java7 -Xmx1G -jar ${GATK_JAR} \
    -T CallableLoci  -R $REF/b37.fasta \
    -o alignment/${i}/${i}.callable.bed \
    --summary alignment/${i}/${i}.callable.summary.txt \
    -I alignment/${i}/${i}.sorted.dup.recal.bam \
    --minDepth 10 --maxDepth 1000 --minDepthForLowMAPQ 10 \
    --minMappingQuality 10 --minBaseQuality 15 \
    -L 19 &
done
wait
```

Now that we have this let's remove/intersect all the samples callable region from the whole genome

```
mkdir substitutions/
cd substitutions/

cat $REF/b37.dict | perl -n -e \
  'if(/^@SQ/){my ($chr,$pos) = /.*SN:([^\t]+)\t.*LN:([0-9]+)\t.*/; print $chr."\t0\t".$pos."\n"}' \
  | grep -v GL | tail -n+2 > wholeGenomeTrack.bed
cp wholeGenomeTrack.bed tmp.bed
for i in ../alignment/*/*.callable.bed
do
  echo $i
  grep "POOR_MAPPING_QUALITY\|CALLABLE" $i |grep -v "^GL" > sampleTmp.bed
  bedtools intersect -a tmp.bed -b sampleTmp.bed > tmp2.bed
  mv tmp2.bed tmp.bed
done
rm sampleTmp.bed
mv tmp.bed projectCallableRegions.bed
cat projectCallableRegions.bed | awk '{SUM+=$3-$2} END {print SUM}'
```

What are the main causes of lost of callability? Solution[49]

Now let's get the rest of the tracks per region. We will extract them from snpEffs database

```
# Remove track header
java7 -Xmx1G -jar $BVATOOLS_JAR extractSnpEffTracks -c $SNPEFF_HOME/snpEff.config hg19 | sed '/^track

awk '{print > "snpEff.hg19."_$4_".bed"}' snpEffRegions.bed
```

Now we have one bed track per region. But we are missing one, do you know which? [Solution)[https://github.com/lletourn/Workshops/blob/ebiCancerWorkshop201407/solutions/__subst.ex2.md)

```
cp wholeGenomeTrack.bed tmp.bed
for i in snpEff.hg19.[UCD]*.bed snpEff.hg19.INTRON.bed
do
  bedtools subtract -a tmp.bed -b $i > tmp2.bed
  mv tmp2.bed tmp.bed
 done
mv tmp.bed snpEff.hg19.INTERGENIC.bed
```

Let's check the size of each region from our project

[49] https://github.com/lletourn/Workshops/blob/ebiCancerWorkshop201407/solutions/__subst.ex1.md

```
for i in snpEff.hg19.*.bed
do
  echo $i
  ORG=‘cat $i | awk ’{SUM+=$3-$2} END {print SUM}’‘
  bedtools intersect -a $i -b projectCallableRegions.bed > callable.${i%.bed}.bed
  CALL=‘cat callable.${i%.bed}.bed | awk ’{SUM+=$3-$2} END {print SUM}’‘
  echo $ORG
  echo $CALL
  echo "$(($CALL*100/$ORG))%"
done
```

Again, here since we down sampled the representation is pretty bad. But in normal WGS you might still find that some regions, like UTR5’ are quite lower than expected. This is were kits like the no PCR help quite a bit.

Now the good part, let’s extract the counts

```
java7 -Xmx5G -jar ~/bvatools-dev.jar mutationrates --vcf ../pairedVariants/mpileup.vcf \
  --bed Callable:projectCallableRegions.bed \
  --bed CDS:snpEff.hg19.CDS.bed \
  --bed DOWNSTREAMcallable.:snpEff.hg19.DOWNSTREAM.bed \
  --bed INTRON:callable.snpEff.hg19.INTRON.bed \
  --bed UPSTREAM:callable.snpEff.hg19.UPSTREAM.bed \
  --bed UTR5:callable.snpEff.hg19.UTR5.bed \
  --bed INTERGENIC:callable.snpEff.hg19.INTERGENIC.bed \
  --bed UTR3:callable.snpEff.hg19.UTR3.bed \
  --minQual 70 --minCLR 45 --threads 1 \
  --exclude MT,GL000207.1,GL000226.1,GL000229.1,GL000231.1,GL000210.1,GL000239.1,GL000235.1,GL000201.
  --output all.hg19.callable.tsv
```

Now open the file in excel, and plot the Base substitution percent histogram.

Finding contamination

This is more to QC but it can be very helpful to find strange patterns in your samples.

Extract positions of somatic variants

```
grep -v INDEL pairedVariants/mpileup.vcf \
 | perl -ne ’my @values=split("\t"); my ($clr) = $values[7] =~ /CLR=(\d+)/; if(defined($clr) && $clr
 > pairedVariants/mpileup.snpPos.tsv
```

Now we have our positions, we need the read counts *per lane* for these positions. BVATools does this

```
for i in normal tumor
do
  java7 -Xmx2G -jar $BVATOOLS_JAR basefreq \
    --pos pairedVariants/mpileup.snpPos.tsv \
    --bam alignment/${i}/${i}.sorted.dup.recal.bam \
    --out alignment/${i}/${i}.somaticSnpPos \
    --perRG
done
```

We can look at one of the files to see what basefreq extracted

```
less -S alignment/normal/normal.somaticSnpPos.normal_C0LWRACXX_1.alleleFreq.csv
```

Now we need to extract and format the data so we can create a PCA and some hierarchical clusters

```
# Generate a part of the command
for i in alignment/*/*.somaticSnpPos*_?.alleleFreq.csv
do
  NAME=`echo $i | sed 's/.*somaticSnpPos.\(.*\).alleleFreq.csv/\1/g'`
  echo "--freq $NAME $i";done | tr '\n' ' '
done

# Copy this output and paste it at the end of the command like so
java7 -Xmx2G -jar ~/bvatools-dev.jar clustfreq \
--snppos pairedVariants/mpileup.snpPos.tsv \
--threads 3 \
--prefix sampleComparison \
--outputFreq \
--freq normal_C0LWRACXX_1 alignment/normal/normal.somaticSnpPos.normal_C0LWRACXX_1.alleleFreq.csv \
--freq normal_C0LWRACXX_6 alignment/normal/normal.somaticSnpPos.normal_C0LWRACXX_6.alleleFreq.csv \
--freq normal_C0PTAACXX_6 alignment/normal/normal.somaticSnpPos.normal_C0PTAACXX_6.alleleFreq.csv \
--freq normal_C0PTAACXX_7 alignment/normal/normal.somaticSnpPos.normal_C0PTAACXX_7.alleleFreq.csv \
--freq normal_C0PTAACXX_8 alignment/normal/normal.somaticSnpPos.normal_C0PTAACXX_8.alleleFreq.csv \
--freq normal_C0R2BACXX_6 alignment/normal/normal.somaticSnpPos.normal_C0R2BACXX_6.alleleFreq.csv \
--freq normal_C0R2BACXX_7 alignment/normal/normal.somaticSnpPos.normal_C0R2BACXX_7.alleleFreq.csv \
--freq normal_C0R2BACXX_8 alignment/normal/normal.somaticSnpPos.normal_C0R2BACXX_8.alleleFreq.csv \
--freq normal_D0YR4ACXX_1 alignment/normal/normal.somaticSnpPos.normal_D0YR4ACXX_1.alleleFreq.csv \
--freq normal_D0YR4ACXX_2 alignment/normal/normal.somaticSnpPos.normal_D0YR4ACXX_2.alleleFreq.csv \
--freq tumor_BC0TV0ACXX_8 alignment/tumor/tumor.somaticSnpPos.tumor_BC0TV0ACXX_8.alleleFreq.csv \
--freq tumor_C0LVJACXX_6 alignment/tumor/tumor.somaticSnpPos.tumor_C0LVJACXX_6.alleleFreq.csv \
--freq tumor_C0PK4ACXX_7 alignment/tumor/tumor.somaticSnpPos.tumor_C0PK4ACXX_7.alleleFreq.csv \
--freq tumor_C0PK4ACXX_8 alignment/tumor/tumor.somaticSnpPos.tumor_C0PK4ACXX_8.alleleFreq.csv \
--freq tumor_C0R29ACXX_7 alignment/tumor/tumor.somaticSnpPos.tumor_C0R29ACXX_7.alleleFreq.csv \
--freq tumor_C0R29ACXX_8 alignment/tumor/tumor.somaticSnpPos.tumor_C0R29ACXX_8.alleleFreq.csv \
--freq tumor_C0TTBACXX_3 alignment/tumor/tumor.somaticSnpPos.tumor_C0TTBACXX_3.alleleFreq.csv \
--freq tumor_D114WACXX_8 alignment/tumor/tumor.somaticSnpPos.tumor_D114WACXX_8.alleleFreq.csv
```

Now you should have 2 files sampleComparison.freq.csv sampleComparison.dist.csv

One contains vectors of snp frequences, the other contains the pairwise Euclidean distance Now plot the result in R

```
fileName <- "sampleComparison"
normalName <- "normal"


distFile <- paste(fileName, ".dist.csv",sep="")

#distName.noext = sub("[.][^.]*$", "", distName, perl=TRUE)
```

```
dataMatrix <- read.csv(distFile, row.names=1, header=TRUE)
hc <- hclust(as.dist(dataMatrix));
hcd = as.dendrogram(hc)

colLab <- function(n) {
    if (is.leaf(n)) {
        a <- attributes(n)
        labCol = c("blue");
        if(grepl(normalName, a$label)) {
          labCol = c("red");
        }
        attr(n, "nodePar") <- c(a$nodePar, lab.col = labCol)
    }
    n
}

# using dendrapply
clusDendro = dendrapply(hcd, colLab)
cols <- c("red","blue")


freqFile <- paste(fileName, ".freq.csv",sep="")
data <- read.csv(freqFile, header=FALSE,row.names=1, colClasses=c("character", rep("numeric",4))
colLanes <- rownames(data)
colLanes[grep(normalName, colLanes, invert=TRUE)] <- "blue"
colLanes[grep(normalName, colLanes)] <- "red"
pca <- prcomp(data)

# make plot
pdfFile <- paste(fileName,".laneMix.pdf", sep="")
pdfFile
pdf(pdfFile)
par(mar=c(3,3,1,12))
plot(clusDendro, main = "Lane distances", horiz=TRUE)
legend("top", legend = c("Normal","Tumor"), fill = cols, border = cols)

par(mar=c(1,1,1,1))
plot(pca$x[,1:2])
text(pca$x[,1:2], rownames(data), col=colLanes)
screeplot(pca, type="lines")
plot(pca$x[,2:3])
text(pca$x[,2:3], rownames(data), col=colLanes)
dev.off()

pngFile <- paste(fileName,".laneMix.png", sep="")
pngFile
png(pngFile, type="cairo", width=1920, height=1080)
par(mar=c(3,3,1,12))
par(mfrow=c(2,1))
plot(clusDendro, main = "Lane distances", horiz=TRUE)
legend("top", legend = c("Normal","Tumor"), fill = cols, border = cols)
```

```
plot(pca$x[,1:2])
text(pca$x[,1:2], rownames(data), col=colLanes)
dev.off()
```

Look at the graphs.

You could do this directly in R but 1- The basefreq format is not simple to parse 2- When you have thousands of somatics, and/or hundreds of samples, R struggles to build de pairwise distance and the PCA. This is why we precompute it in java before.

Aknowledgments

The format for this tutorial has been inspired from Mar Gonzalez Porta of Embl-EBI, who I would like to thank and acknowledge. I also want to acknowledge Mathieu Bourgey, Francois Lefebvre, Maxime Caron and Guillaume Bourque for the help in building these pipelines and working with all the various datasets.