BCGES short courses, session 1
Picard tools, CNV analysis

Vincent Plagnol

1 Two more advanced ideas to consider

1.1 `samtools` over the web to download BAM files

A powerful feature of BAM/SAM and `samtools` is the ability to use the `view` feature with indexes over the web, to only pull the chunks of interest. I will provide an example of this using the 1,000 Genomes data. To do that, one should first download the list of whole genome alignments from the 1,000 Genomes to identify where the files of interest are located on the server.

```
wget -O 1KG_alignment.index ftp://ftp.1000genomes.ebi.ac.uk/vol1/ftp/alignment.index
```

Exercise: Download data from a BAM file generated by the ENCODE project that contains ChIP-Seq information about *GATA2* binding sites 1 Mb around the gene *IL18RAP*. Just to help the search look for the data located here.

1.2 Linux piping to limit input/output usage

The lines of code below cannot be run as such but are just meant to give an example of the sort of things that multiple lines of code put together can do. In this example the only thing written to the disk is the sorted BAM file. No other intermediate file is written to disk.

```
novoalign -c 11 -o SAM -F STDFQ -f fasta1.fq fasta2.fq  -d $reference |
   samtools view - -u -S -b |
   novosort - -t /scratch0/ -c 1 -m 3G -i -o output_sorted.bam
```

Exercise: Understand what the various options mean, at least for samtools which is freely available.

2 Some examples PICARD tools

2.1 Marking duplicates

```
BAM=../data/BAM_files/HG00130.mapped.ILLUMINA.bwa.GBR.exome.20130415.bam
java -Xmx4g -jar picard/MarkDuplicates.jar  ASSUME_SORTED=true \
      REMOVE_DUPLICATES=FALSE \
      INPUT=$BAM \
      OUTPUT=HG00130.mapped.bam METRICS_FILE=HG00130.metrics.out
```

2.2 Collect summary statistics

And here is some example code to compute summary statistics on a BAM file:

```
BAM=../data/BAM_files/HG00130.mapped.ILLUMINA.bwa.GBR.exome.20130415.bam
java -Xmx4g -jar picard/CollectAlignmentSummaryMetrics.jar \
    INPUT=$BAM \
    OUTPUT=insert_size.txt
```

2.3 Convert a BAM file to fastq

Now something quite a bit more challenging: try the following script to convert a BAM file to fastq: Here is a first attempt

```
BAM=../data/BAM_files/HG00130.mapped.ILLUMINA.bwa.GBR.exome.20130415.bam
java -Xmx4g -jar picard/SamToFastq.jar \
    INPUT=$BAM \
    FASTQ=read1.fq SECOND_END_FASTQ=read2.fq
```

The issue is that some reads do not have a mate, and that creates issues with the FASTQ files. We need a fix to deal with that error message. We can try the following, which should remove non-mapped reads:

```
BAM=../data/BAM_files/HG00130.mapped.ILLUMINA.bwa.GBR.exome.20130415.bam
samtools view -f 0x0001 -f 0x0002 -b -o only_paired.bam $BAM

java -Xmx4g -jar picard/SamToFastq.jar INPUT=only_paired.bam FASTQ=read1.fq SECOND_END_FASTQ=read2.f
```

But it fails again. With only 2 unpaired mates this time, so things are getting better. We can find out what these are:

```
samtools view only_paired.bam | awk '{print $1}' | sort | uniq -c | sort -r | tail -5

##       2 SRR707198.10122
##       2 SRR707198.10048168
##       2 SRR707198.10047979
##       1 SRR707198.24062323
##       1 SRR707198.20187117
```

Now let us remove these problematic reads, but we have to do it manually. I could not find a better way to do this than what is shown below:

```
samtools view -h only_paired.bam |
       awk '{if (($1 != "SRR707198.24062323") && ($1 != "SRR707198.20187117")) print}' > paired_fixe
samtools view -b -S -o paired_fixed.bam paired_fixed.sam
```

The second line above takes a SAM file input (option -S) and returns a BAM file (option -b). The resulting file can, at last, be converted into a FASTQ format:

```
java -Xmx4g -jar picard/SamToFastq.jar INPUT=paired_fixed.bam FASTQ=read1.fq  SECOND_END_FASTQ=read2
```

Exercise: Can you find a better way to convert a BAM to fastq? I could not but I have not looked extensively.

3 Read depth analysis to detect CNVs

Run the example code provided in `exomeDepth_example.R`. Familiarise yourself with the code, modify it to run with the sample `UCLG_94_sorted_unique.bam`. You should be able to see a two exon heterozygous deletion located in the gene *GATA2*, which is causal in this individual.

```
library(ExomeDepth)
```

```
ExomeCount <- read.table('../data/exon_read_count.tab.gz', header = TRUE)
my.test <- ExomeCount$UCLG.186_sorted_unique.bam
all.bams <- grep(pattern = 'bam$', names(ExomeCount), value = TRUE)
my.ref.samples <- subset( all.bams, all.bams != 'UCLG.186_sorted_unique.bam')

my.reference.set <- as.matrix(ExomeCount[, my.ref.samples])

my.choice <- select.reference.set (test.counts = my.test,
                                   reference.counts = my.reference.set,
                                   bin.length = (ExomeCount$end - ExomeCount$start)/1000,
                                   n.bins.reduced = 10000)

## Warning:  The data set contains at least one line with weight = 0.
##
## Warning:  The data set contains at least one line with weight = 0.
##
## Warning:  The data set contains at least one line with weight = 0.
##
## Warning:  The data set contains at least one line with weight = 0.
##
## Warning:  The data set contains at least one line with weight = 0.
##
## Warning:  The data set contains at least one line with weight = 0.
```

```
my.matrix <- as.matrix( ExomeCount[, my.choice$reference.choice, drop = FALSE])
my.reference.selected <- apply(X = my.matrix,
                               MAR = 1,
                               FUN = sum)

######### create the ExomeDepth object
all.exons <- new('ExomeDepth',
                 test = my.test,
                 reference = my.reference.selected,
                 formula = 'cbind(test, reference) ~ 1')

## Warning:  The data set contains at least one line with weight = 0.
```

```
all.exons <- CallCNVs(x = all.exons,
                      transition.probability = 10^-4,
                      chromosome = ExomeCount$chromosome,
                      start = ExomeCount$start,
                      end = ExomeCount$end,
                      name = ExomeCount$exons)


################## Now annotate the CNV calls
data(Conrad.hg19)
all.exons <- AnnotateExtra(x = all.exons,
```

```
                           reference.annotation = Conrad.hg19.common.CNVs,
                           min.overlap = 0.5,
                           column.name = 'Conrad.hg19')

data(exons.hg19)

exons.hg19.GRanges <- GRanges(seqnames = exons.hg19$chromosome,
                              IRanges(start=exons.hg19$start,end=exons.hg19$end),
                              names = exons.hg19$name)

all.exons <- AnnotateExtra(x = all.exons,
                           reference.annotation = exons.hg19.GRanges,
                           min.overlap = 0.0001,
                           column.name = 'exons.hg19')



####### Take the final table, sort by significance (Bayes factor)
my.final.table <- all.exons@CNV.calls
my.final.table <- my.final.table[ order(my.final.table$BF, decreasing = TRUE),]



print(table(is.na(my.final.table$Conrad.hg19)))

##
## FALSE  TRUE
##    33    18

print(head(my.final.table))

##    start.p end.p        type nexons     start       end chromosome
## 51   66663 66693    deletion     31 151749334 151948710          4
## 17   11958 11982    deletion     25 104202984 104293260          1
## 8     2813  2821    deletion      9  16905690  16915559          1
## 23   13791 13795    deletion      5 143361979 143375274          1
## 3     2273  2287 duplication     15  13109020  13165444          1
## 4     2292  2295    deletion      4  13196272  13217250          1
##                          id    BF reads.expected reads.observed
## 51 chr4:151749334-151948710 913.0           5348              8
## 17 chr1:104202984-104293260 129.0           1025            196
## 8    chr1:16905690-16915559 114.0           6259           3302
## 23 chr1:143361979-143375274  72.4            550            115
## 3    chr1:13109020-13165444  66.0            510           1255
## 4    chr1:13196272-13217250  48.0            322             50
##    reads.ratio         Conrad.hg19
## 51      0.0015               <NA>
## 17      0.1910 CNVR266.3,CNVR266.5
## 8       0.5280   CNVR93.3,CNVR93.6
## 23      0.2090          CNVR330.2
## 3       2.4600   CNVR73.1,CNVR73.2
## 4       0.1550   CNVR73.2,CNVR73.3
##
## 51 LRBA_30,LRBA_29,LRBA_28,LRBA_27,LRBA_26,LRBA_25,LRBA_24,LRBA_23,LRBA_22,LRBA_21,LRBA_20,LRBA_1
## 17                                                                                              AM
## 8
```

```
## 23
## 3
## 4

pdf('LRBA_example.pdf')
plot (all.exons,
      sequence = '4',
      xlim = c(151749334 - 100000, 151948710 + 100000),
      count.threshold = 20,
      main = 'LRBA gene',
      with.gene = TRUE)

dev.off()

## pdf
##   2
```

4 Use `Rsamtools` to identify reads characteristic of a deletion

Another strategy to detect CNVs consists of picking up reads that show an unusual pattern. For example, two pairs further apart than they normally should is potentially flagging a deletion. In this exercice I proposed to look at a well described 20 kb CNV located near the *IRGM* gene. We can start by looking at the location of the variant using the data from the Conrad et al paper, which are loaded into ExomeDepth.

```
data(Conrad.hg19)
IRGM <- Conrad.hg19.common.CNVs[ seqnames(Conrad.hg19.common.CNVs) == 5 &
                                 start(Conrad.hg19.common.CNVs) > 150200000
                                 & end(Conrad.hg19.common.CNVs) < 150240000,]
write.table(x = as.data.frame(IRGM),
            row.names = FALSE, sep = '\t', quote = FALSE,
            file = 'IRGM_common.tab')
```

We now want to look for individuals that carry this variant. Low coverage whole genome data for the 1KG sample HG00123 will do the job.

Exercise: Identify the BAM file for mapped reads for HG00123 and download the slice 5:150200000-150225000 using `samtools view`. How would you filter pairs of reads that span over the deletion in this sample?

(Optional) Exercise: Can you write a R script (using Rsamtools) to identify these reads?