BCGES short courses, session 5: Picard tools, CNV analysis, BEDtools

Vincent Plagnol

Contents

1 A few more advanced ideas to consider (40 minutes)

1.1 Linux piping to limit input/output usage (10 minutes)

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 the structure of the code and what the options used in the code above mean (at least for samtools which is freely available).

1.2 CRAM format as an alternative to BAM (10 minutes)

Recently (and after we finished preparing the software for this course), the `samtools` developers released version 1.0 of the software. A key addition is the handling of the CRAM format, an optimized format that compresses BAM further. Given the very large size of exome/genome datasets, CRAM has the potential to make a real difference. The version 1.0 of `samtools` has been installed in the bin subdirectory of session 5, and hopefully runs without any issue on your current machine. The main page for `samtools` has moved to a different location, and the associated documentation is here. I strongly recommend reading this very good blog post on the CRAM format that will tell you more about why this is useful.
Exercise: Take an example BAM file provided as part of this practical and convert it to CRAM. Do you see a reduction of the file size?

1.3 Tabix indexing of position sorted files (20 minutes)

TO BE WRITTEN

2 Some examples PICARD tools (40 minutes)

2.1 Marking duplicates

Here is an example below of a typical PICARD call to mark duplicates. Make sure that you can run some version of this script, this is a very standard process to go through.

```
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=results/HG00130.mapped.bam METRICS_FILE=results/HG00130.metrics.out
```

Exercise: Can you find where, in the headers of the resulting BAM file, how information is recorded about this duplicate marking step?

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=results/insert_size.txt
```

Make sure that you can run that code, and look at the output files. Check that you understand the meaning of the summary statistics.

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=results/read1.fq SECOND_END_FASTQ=results/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 results/only_paired.bam $BAM

java -Xmx4g -jar picard/SamToFastq.jar INPUT=results/only_paired.bam FASTQ=results/read1.fq SECOND_E
```

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 results/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 results/only_paired.bam |
       awk '{if (($1 != "SRR707198.24062323") && ($1 != "SRR707198.20187117")) print}' > results/pai
samtools view -b -S -o results/paired_fixed.bam results/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=results/read1.fq  SECOND_END_FAS
```

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

3 BEDtools (30 minutes)

BEDtools utilities are a swiss-army knife of tools for a wide-range of genomics analysis tasks. Individually, none of these tools looks impressive, but together with `samtools` they can be used to achieve impressive tasks. I could simply point you to the excellent documentation of the software (and you certainly should have a look) but I will also highlight a few useful ideas.

3.1 Computing depth of coverage for targeted DNA sequencing experiments

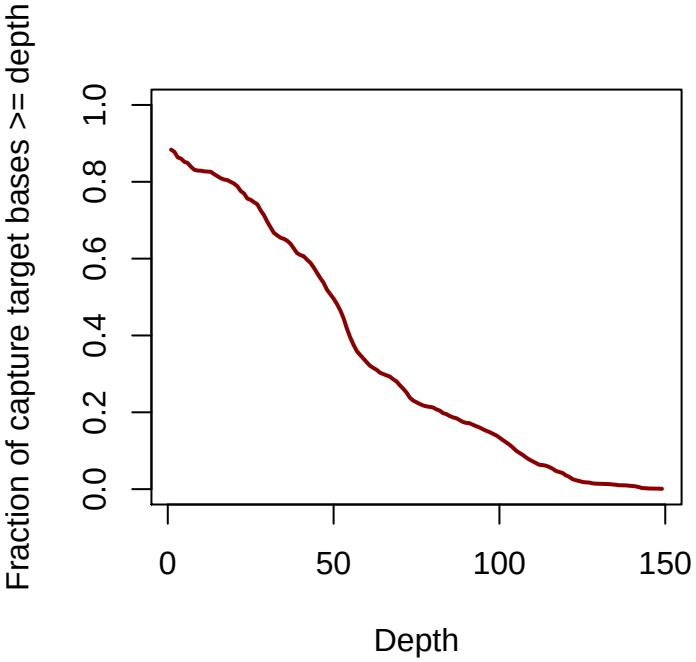
Some fancy bits of code are proposed in that blog post. We will simply follow these ideas to illustrate what `BEDtools` can do. The first step is to actually obtain a BED file that describes where the exons of a gene of interest are located. I generated this using this very good post, and if there is time I suggest that you look into this. My modified scripts (which worked in this case) are located in `tables/join.sh`. **Optional exercise:** Reproduce the steps suggested in the Biostars answer. Scripts will need to be updated slightly depending on the exact computer that you use.

This is however not the main topic of the day and feel free to simply use the outcome BED file `tables/UBASH3A.bed`.

```
BAM=../data/BAM_files/HG00130.mapped.ILLUMINA.bwa.GBR.exome.20130415.bam
bedtools coverage -hist -abam $BAM -b tables/UBASH3A.bed | grep ^all > results/HG00130_coverage.txt
head results/HG00130_coverage.txt

## all 0 284 2442 0.1162981
## all 1 14 2442 0.0057330
## all 2 35 2442 0.0143325
## all 3 8 2442 0.0032760
## all 4 20 2442 0.0081900
## all 5 8 2442 0.0032760
## all 6 24 2442 0.0098280
## all 7 19 2442 0.0077805
## all 8 5 2442 0.0020475
## all 9 1 2442 0.0004095
```

What you get from this is the data underlying an histogram. The fourth column tells you that you are looking at a total of 2,442 bp. For the first row, the third column tells you 284 of these positions are covered by 0 read, 14 are covered with 1 read, 35 positions with 2 reads... This is not very readable, so the output can for example be processed in `R` using the code below:

```
gcov = read.table('results/HG00130_coverage.txt');

# Create a cumulative distribution from the "raw" hist
# (truncate at depth >=1000)
gcov_cumul = 1 - cumsum(gcov[,5])

## Create a plot of the CDF
plot(x = gcov[2:143,2], y = gcov_cumul[1:142],
     col='darkred', type='l', lwd=2,
     xlab="Depth",
     ylab="Fraction of capture target bases >= depth",
     ylim=c(0,1.0))
```



Exercise: Make sure you can run and understand the `R` code, and what the different columns of the output mean. Can you rewrite the code above so that it loops over all the GBR BAM files? You will probably need the function `find` to do this. You can also try the fancier looking `parallel` implementation proposed by Stephen Turner's blog post but be careful, `parallel` is not systematically installed on all machines so thuis may not be possible.

3.2 Find regions that are not covered by a BAM file

This originated from the following question, asked on Twitter: "Given a.bam and b.regions.bed, how to get the parts of b.regions.bed that are not covered by a.bam?" See the example below:

```
BAM=../data/BAM <- files/HG00130.mapped.ILLUMINA.bwa.GBR.exome.20130415.bam
regions=tables/UBASH3A.bed
bedtools genomecov -ibam $BAM -bga \
         | awk '$4==0' | bedtools intersect -a $regions -b - > results/not_covered.bed
```

Question: Understand what the three piping steps do in this example. Look at the output. How many bases are not covered? Is this consistent with the result file of the previous section?

4 Use `Rsamtools` to identify reads characteristic of a deletion (30 minutes)

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. These are loaded into the ExomeDepth package, so this is one way to access them.

```
library(ExomeDepth)

## Loading required package:  methods
## Loading required package:  aod
## Loading required package:  VGAM
## Loading required package:  splines
## Loading required package:  stats4
## Loading required package:  GenomicRanges
## Loading required package:  BiocGenerics
## Loading required package:  parallel
##
## Attaching package:  'BiocGenerics'
##
## The following objects are masked from 'package:parallel':
##
##    clusterApply, clusterApplyLB, clusterCall, clusterEvalQ,
##    clusterExport, clusterMap, parApply, parCapply, parLapply,
##    parLapplyLB, parRapply, parSapply, parSapplyLB
##
## The following object is masked from 'package:stats':
##
##    xtabs
##
## The following objects are masked from 'package:base':
##
##    anyDuplicated, append, as.data.frame, as.vector, cbind,
##    colnames, duplicated, eval, evalq, Filter, Find, get,
##    intersect, is.unsorted, lapply, Map, mapply, match, mget,
##    order, paste, pmax, pmax.int, pmin, pmin.int, Position, rank,
##    rbind, Reduce, rep.int, rownames, sapply, setdiff, sort,
##    table, tapply, union, unique, unlist
##
## Loading required package:  IRanges
## Loading required package:  XVector
## Loading required package:  Rsamtools
## Loading required package:  Biostrings

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?
Exercise: Use IGV to visualize the deletion and make sure you can clearly see these reads.
(Optional) Exercise: Can you write a R script (using `Rsamtools`) to identify these reads?