A Copy Number Analysis Pipeline

Rish Prabakar

January 14, 2024

1 Introduction

This document is intended to serve as a walk through of the copy number analysis pipeline.

Assumptions:

1. A familiarity with the UNIX computing environment.
2. The ability to compile and install tools and packages.
3. A familiarity with the commonly used bioinformatic file types such as fasta, fastq, and sam/bam.

A personal workstation with at least 16GB of memory and 5GB of free disk space can be used for the analysis of one or at most a few samples. For processing a large number of samples, it is highly recommended that the analysis is done on a compute cluster. A compute cluster provides hundreds of nodes that can process files in parallel, and thus significantly speeding up the analysis.

2 Copy number analysis

This copy number analysis pipeline is based on the procedure described in [Baslan et al., 2012, Kendall and Krasnitz, 2014], with modifications to bin boundaries that are designed for 150bp paired-end reads, and with an additional step of explicitly filtering reads that align to ambiguous or problematic regions of the reference genome. Briefly, the human reference genome (hg19 or hg38) is split into bins (typically 5000, but can higher or lower depending on the resolution needed and number of reads sequenced) containing an equal number of uniquely mappable locations, and the bin counts are determined using uniquely mapped reads that do not align to the ambiguous or problematic regions. Bins with spuriously high counts ("bad bins," typically around centromeric and telomeric regions) are masked for downstream analysis. This procedure normalizes bin counts for biases correlated with GC content by fitting a LOWESS curve to the GC content by bin count, and subtracting the LOWESS estimate from each bin. Circular binary segmentation (CBS) [Olshen et al., 2004], then identifies breakpoints in the normalized bin counts.

2.1 Tools and packages required

The following tools and packages are required for copy number analysis:

1. FastQC

2. bwa
3. samtools
4. picard tools (requires Java, unfortunately)
5. Python ($\geq$ 3.0)
6. Python packages: pysam
7. R ($\geq$ 4.0)
8. R packages: optparse and DNAcopy
9. Optional: MultiQC

All tools are assumed to the located in a directory specified in `$PATH` and all files are assumed to be in the current working directory. If they are not, the commands below can be modified to include the absolute or relative paths to the tool and required files.

2.2 Preparing the reference genome

hg19 download and pre-processing: The hg19 reference genome can be downloaded from the UCSC Genome Browser. Alternate haplotypes are not used for analysis. Additionally, the pseudoautosomal regions on chrY are masked by replacing the bases with Ns. All the chromosome files can then the merged into one fasta file.

hg38 download: The hg38 reference genome can be downloaded from the UCSC Genome Browser. The hg38 "Analysis set" has the PAR regions masked and does not contain alternate haplotypes. No pre-processing is required and the genome can be used directly.

Generating BWA index: `bwa`, the program used to align reads to the reference genome, requires the reference genome to be prepossessed once before the first use.

```
$ bwa index {reference.fa}
```

2.3 Pre-alignment QC

Prior to any analysis it is crucial to QC the fastq files to detect any errors in the library preparation or the sequencing process. `FastQC` provides a set of diagnostic plots on fastq files.

```
$ fastqc read_1.fastq.gz read_2.fastq.gz -o fastqc_outdir
```

The output of `FastQC` is one HTML file per fastq file containing some basic statistics and a set of diagnostic plots. These HTML files can be view on any browser. An explanation of these plots are provided here. Some important metrics to look for are the read length distribution, sequence quality, over-represented sequences such as sequencing adapters, and GC content.

It is generally inconvenient to view one HTML file for every fastq file. The tool `MultiQC` can be used to aggregate all the information for all the samples into one HTML file.

2.4 Align reads to the reference genome

The first step in processing the reads is to align them to the reference genome to determine their location on the reference genome. The reads are aligned to the reference genome using `bwa` [Li, 2013]. The input to `bwa` are a pair of fastq files and the pre-built reference index (as described in section 2.2).

The input reads usually contain primer sequences from whole genome amplification or adapter sequences from the Illumina library preparation process at the ends. `bwa` performs a local alignment on the reads and soft clips any of these unwanted technical bases that would not align to the reference genome. Therefore, is not required to explicitly trim the adapter sequences prior to processing with `bwa`.

```
$ bwa mem -t {threads} {path_to_bwa_index} \
    read_1.fastq.gz read_2.fastq.gz \
    1> sample.sam 2> sample_bwa_log.txt
```

The output of `bwa` is a sam file. The first few lines of a sam file contain a header that starts with `@`. The subsequent lines contain one entry for each read, and thus two entries for a read pair (but could contain more than two entries when a read is split and aligned to more than one location on the reference). Each entry is tab delimited, and the first five columns provide the most useful information in the context of copy number profiling: (1) The read name. (2) A decimal representation of a bitwise combination of 12 bits, where each bit represents a property of the alignment. (3) The chromosome to which the read aligns to. (4) The position on the chromosome to which the read aligns to. (5) Mapping quality of the read which is calculated as $-10\log_{10}\Pr(\text{mapping position is wrong})$. A detailed specification of a sam file format can be found here.

Sam files are convenient to view on a text editor. However, these files consume a lot of disk space since they are not compressed. Bam files are the compressed version of sam files, and the formats can be converted with `samtools` (the sam file can then be deleted).

```
$ samtools view -@ {threads} -b -o sample.bam sample.sam
```

2.5 Remove PCR duplicates

There are several steps in the process of going from $\sim$6pg of DNA in a cell to the $\sim$100ng required for Illumina library preparation involves PCR amplification, which could lead to capturing two identical DNA fragments in the sequencing process. Since these reads are identical, the presence of more than one read pair for identical fragments does not provide any additional information, but could lead to a bias in the downstream analysis.

PCR duplicates are removed based on the assumption that two read pairs that align to exactly the same location on the reference genome are more likely due to a PCR duplicate rather than two different fragments. This is a reasonable assumption for whole genome sequencing at a low coverage especially for a single cell. (However, this method of removing PCR duplicates cannot be used for high coverage or targeted sequencing.) Multiple read pairs that align to the reference at the same 5' location for both the forward and the reverse reads are considered as PCR duplicates. A graphical representation of a PCR duplicate can be found here.

A good library that has a high complexity contains very few duplicated reads. A library that contains a high fraction of PCR duplicates ($> 30 - 40\%$) could be indicative of low DNA content in the cell or high genome drop out, and should be treated with caution.

PCR duplicates are removed with `samtools` with a set of four commands. The first command sorts the alignments in a bam file based on the read names so that all the alignments from the same read pair are adjacent in the bam file.

```
$ samtools collate -@ {threads} -o sample_collate.bam sample.bam
```

`bwa` and other mapping tools typically do not provide the correct information about the insert size for the read pairs. `samtools fixmate` fixes these.

```
$ samtools fixmate -@ {threads} -m sample_collate.bam \
    sample_fixmate.bam
```

The bam file is then sorted based on the genome coordinates to facilitate removing PCR duplicates.

```
$ samtools sort -@ {threads} -o sample_sorted.bam sample_fixmate.bam
```

Finally, the PCR duplicates are removed and only the best alignment for each set of duplicates is retained.

```
$ samtools markdup -@ {threads} -r sample_sorted.bam sample_rmdup.bam
```

The final output of these steps is a bam file in which all the duplicate reads are discarded, and thus contains fewer alignments than the input bam file.

2.6 Remove low quality and ambiguously mapped reads

Reads that align to several different locations on the reference genome receive a low mapping quality score (in the fifth column of a sam entry). Since a unique location for these alignments cannot be determined, they do not provide any useful information for copy number profiling and they can be discarded. Further, for reads that get split and align to multiple locations, only the best alignment is retained. The alignments are filtered with `samtools`:

```
$ samtools view -@ {threads} -q 30 -F 0x800 -o sample_unique.bam \
    sample_rmdup.bam
```

The output is a bam file that retains only the unique alignments.

2.7 Remove mates

Since a sam file contains two entries for each read pair, one of these needs to be removed prior to copy number profiling. Not doing so would lead to double counting each alignment. `samtools` is used to remove the alignments corresponding to the second read of the read pair.

```
$ samtools view -@ {threads} -f 0x40 -h -o sample_fwd.bam \
    sample_unique.bam
```

The output is a bam file that retains only the alignments that belong to the first read of the read pair.

2.8 Post alignment QC

It is highly recommended to check the number of alignments that are discarded after each step of the above filtering process. The number of alignments, the number of read pairs, etc in a sam file can be determined using the command:

```
$ samtools flagstat {bam_file.bam}
```

Of note, in the output of `samtools flagstat`, the fraction of aligned reads are calculated based on the number of alignments in the sam file (and not based on the number of reads in the fastq files for the sample).

2.9 Insert size distribution

As an additional QC, it is important to check the insert size distribution of the aligned reads. A good sequencing library should have a majority of reads with an insert size greater than 100bp. Samples with a large fraction of inserts less than 50bp should be treated with caution.

```
$ java -jar picard CollectInsertSizeMetrics I=sample_unique.bam \
  O=sample_insert_sz.txt H=sample_insert_sz.pdf
```

2.10 Generate bin counts

At this stage, the sam file is ready to be used for copy number profiling. The next step is to determine the number of reads that align to each predetermined bins on the reference genome. The bin boundaries were determined so that the number of uniquely mappable regions in the bins are approximately the same across all the bins (and so the absolute width of each bin varies, with larger bins in regions of the genome that contain repetitive sequences and smaller in regions that mostly contain unique sequences).

Further, there are certain regions of the genome that are known to be "problematic" that have an unusually high number of unique reads aligning to them [Amemiya et al., 2019]. In addition to the ambiguously mapping regions of the genome, these problematic regions are excluded as well when determining the bin boundaries.

`getBinCounts.py` takes as input a bed file containing the genomic coordinates of the bin boundaries, a bed file containing the genomic coordinates of all the ambiguously mapping regions and the problematic regions, and the sam file containing only the forward reads. For each alignment in the sam file, it is discarded if it is aligned to ambiguous or problematic regions, otherwise the count of the bin it belongs to is incremented by one.

```
$ getBinCounts.py -i sample_fwd.bam \
    -b {hg19/hg38}_{bins}_gz_enc_bins.bed \
    -d {hg19/hg38}_150bp_dz_enc.bed -o sample_5k_counts.bed \
    -v > sample_5k_counts_stats.txt
```

`getBinCounts.py` is located in the scripts directory and the bed files are located in the data directory in the GitHub repository. For example, to generate profiles for hg38 at 5000 bin resolution, use `-b hg38_5k_gz_enc_bins.bed` and `-d hg38_150bp_dz_enc.bed`. For generating profiles for hg19 or at other bin resolutions, use the appropriate bed files.

The output is a bed file that contains the genomic coordinates of the bin boundaries with an added fourth column containing the counts for that bin. The stats file provides some basic statistics on the number of input alignment, the number of filtered alignment, and the number of alignment that is used for copy number profiling.

2.11 Generate copy number profiles

The final step is to convert the bin counts into bin ratios and segmented copy number profiles. The counts are normalized to account for the differences in the number of reads sequenced between samples, LOWESS smoothed to account for the differences in GC content between bins, and segmented using circular binary segmentation.

`cnvProfile.R` takes as input the bin counts, a bed file containing the GC content of each bin, and an optional bed file containing the "bad bins."

```
$ cnvProfile.R -b sample_5k_counts.bed
    -g {hg19/hg38}_{bins}_gz_enc_gc.bed \
    -e {optional:bad_bins_bed} \
    -o {output_dir} -n {sample_name} -v
```

The output is a pdf file containing a copy number profile, and two tab separated files containing all the information related to the copy number profile such as the bin ratios and the segmented values. One of these file contains the same number of rows as the number of bins used, and the other file is shorter version that contains one row for each copy number segment. When the bad bins are specified, another set of these three files are generated with the bad bins removed. The these files can be used as an input to any downstream applications such as generating copy number heatmaps.

References

Haley M Amemiya, Anshul Kundaje, and Alan P Boyle. The encode blacklist: identification of problematic regions of the genome. *Scientific reports*, 9(1):1–5, 2019.

Timour Baslan, Jude Kendall, Linda Rodgers, Hilary Cox, Mike Riggs, Asya Stepansky, Jennifer Troge, Kandasamy Ravi, Diane Esposito, B Lakshmi, et al. Genome-wide copy number analysis of single cells. *Nature protocols*, 7(6):1024–1041, 2012.

Jude Kendall and Alexander Krasnitz. Computational methods for dna copy-number analysis of tumors. In *Cancer Genomics and Proteomics*, pages 243–259. Springer, 2014.

Heng Li. Aligning sequence reads, clone sequences and assembly contigs with bwa-mem. *arXiv preprint arXiv:1303.3997*, 2013.

Adam B Olshen, ES Venkatraman, Robert Lucito, and Michael Wigler. Circular binary segmentation for the analysis of array-based dna copy number data. *Biostatistics*, 5(4):557–572, 2004.