NGS analysis guide - CNV and repeats analysis

This book will outline the analysis workflow of two main analysis: - copy number variation (CNV) analysis - repeat quantitation using repeatmasker

General flow

DNA ➞ NGS ➞ raw read sequence (fastq) ➞ read alignment to reference genome (bam)
➘ repeat discovery

Getting your DNA sequence data

When carrying out cnv analysis, we need a control dataset to compare with out own NGS data.
We will get the publicly available dataset from sra (Sequence read archive) database.

The command used is *fastq-dump*. Fastq-dump fetched data from the sra database. *–split-files* is used when the ngs data is paired-end, to tell the program to separate the reads into two files.
DRR (or SRR) indicate the accession number of the data in the sra database.

```
fastq-dump -I --split-files DRR231927
```

After the data is downloaded, run fastqc to check the quality of the data.

```
fastqc seq_r1.fastq seq_r2.fastq
firefox fastqcreport.html
```

If the data quality is ok, proceed to align the data against the reference genome.

Alignment to reference genome

1. Before starting alignment, reference genome need to be index first. This only need to be done one time for one reference genome (no need to index reference genome for subsequent alignment).

```
bwa index genome.fa
# replace genome.fa with your reference genome
```

2. alignment and sorting of reads according to genomic position of reference genome

The step above generate alignment according to your read sequence (as it is come out from your sequencer), and subsequently sort the aligned reads in correct order as the reference genome (the genome viewer require sorted reads to open the file).

-t : number of threads use to carry out the alignment process
-o : write to file instead on terminal
| : pipe function on command line to direct the output from previous command as input for the next command

```
bwa mem -t 4 genome.fa read1.fastq read2.fastq | samtools sort -o file.bam
```

3. index bam file using samtools
   After sorting, the bam file requires indexing to refer to a read quickly (much like books in a library are sorted as catologue that enable fast retrieval).

```
samtools index yourfile.bam
```

–> Your file is ready to be viewed in genome browser such as IGV.

Copy number variation (CNV) analysis

The copy number variation analysis can only be done after you have two aligned and sorted file (one as control and another is your sample).

There are a lot of program available for cnv analysis. In this case, we will use cnv-seq. You can refer to the cnv-seq wiki page (https://github.com/Bioconductor/copy-number-analysis/wiki/CNV-seq). The following steps will follow closely as in the wiki page.

There are basically two main steps. First, to generate best-hit location based on your aligned ngs reads.
-F 4 means read is mapped
$F[2] is the rname (read name)
$F[3] is the alignment position

The whole command means : get the mapped position from your alignment file (bam), and ask perl Programming language to print the read name and mapped position to the *mapped.hits* file.

```
samtools view -F 4 yourfile.bam | perl -lane 'print "$F[2]\t$F[3]"' > yourfile.mapped.hits
```

Next, the number of mapped reads in each window (eg. number of reads per 1 kb) were counted and log2 ratio were calculated.

```
perl '/mnt/D/cnv-seq-master/cnv-seq.works.hliang.tch.v2.pl' --test yourfile.hits --ref contro
l.hits --genome human --Rexe /usr/bin/R --annotate
```

This will produce two output files. The .cnv file can then be imported into R for plotting and visualisation.

Plotting and visualizing copy number variation

We will mainly use the ggplot package in R for plotting the cnv. (Replace .cnv with your own file)

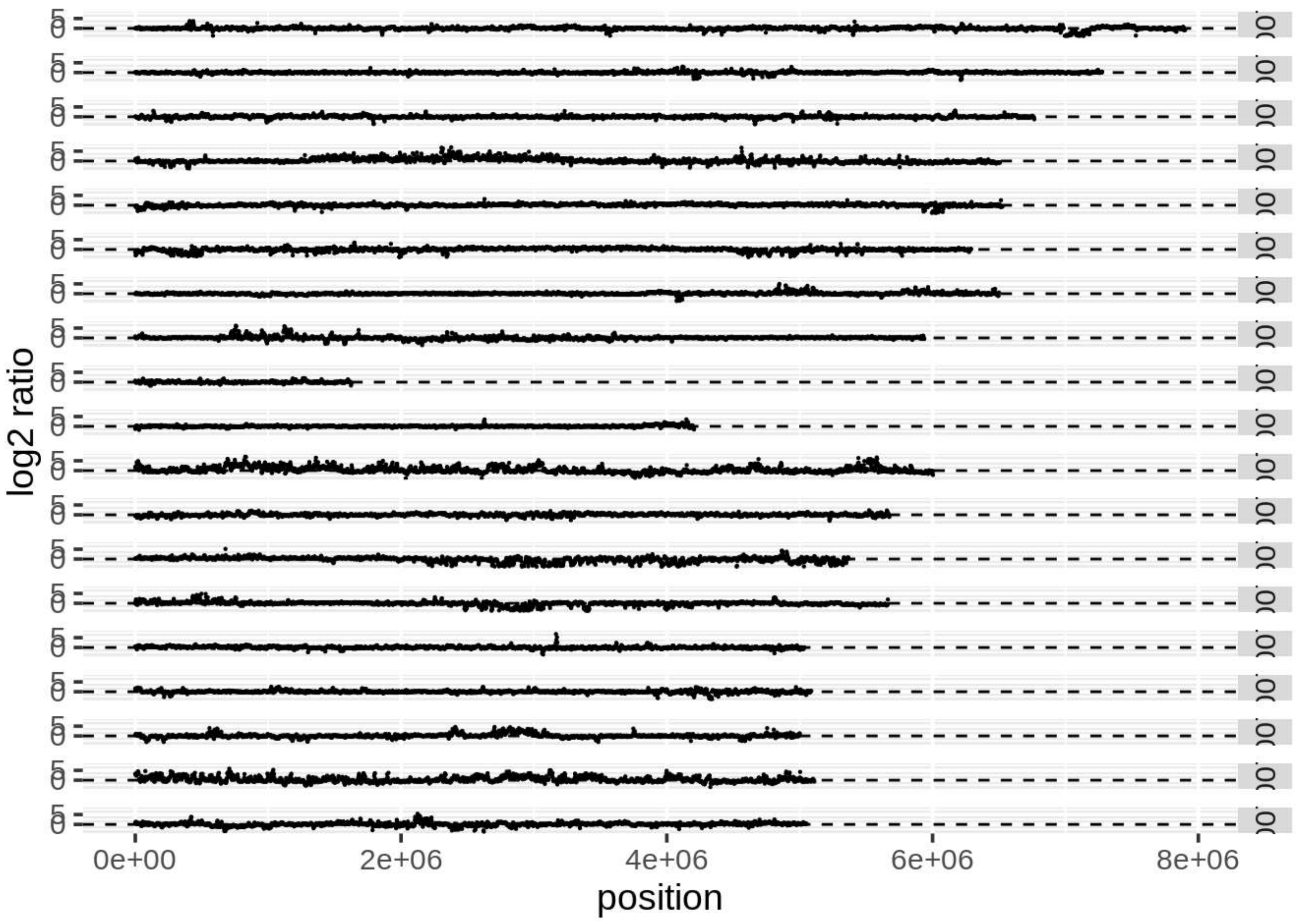
```r
library(ggplot2)
# read the table into R
cnv <- read.table("/home/user/Desktop/hydrangea/Liaw/cnv/hydrangea.mapped.hits-vs-hydrangea_c
ontrol_DRR231917_part.mapped.hits.log2-0.6.pvalue-0.001.minw-4.cnv", sep="\t", header=TRUE, s
tringsAsFactors=FALSE)

# remove data points that has infinity in the log2 ratio
cnv <- cnv[is.finite(cnv$log2), ]

# The draft reference genome contained 3000+ chromosomes, so we will just retain the first 19
largest chromosomes asembled
cnv <- cnv[cnv$chromosome %in% c("BLYA01000001.1", "BLYA01000002.1", "BLYA01000003.1", "BLYA0
1000004.1", "BLYA01000005.1", "BLYA01000006.1", "BLYA01000007.1", "BLYA01000008.1", "BLYA0100
0009.1", "BLYA01000010.1", "BLYA01000011.1", "BLYA01000012.1", "BLYA01000013.1", "BLYA0100001
4.1", "BLYA01000015.1", "BLYA01000016.1", "BLYA01000017.1", "BLYA01000018.1", "BLYA01000019.
1"),]

# plot the data using ggplot. with x-axis = start position and y-axis = log2 ratio
g <- ggplot(cnv,aes(x=start, y=log2))
g + geom_point(size=0.1) + facet_grid(chromosome ~ . ) + geom_hline(yintercept = 0, linetype
 = "dashed", color = "black") + ylim(-5, 10) + theme(axis.ticks = element_blank()) + ylim(-4,
8) + theme_grey(base_size = 14) + ylab("log2 ratio") + xlab("position")
```

```
## Scale for 'y' is already present. Adding another scale for 'y', which will
## replace the existing scale.
```



CNV analysis done

– locate any chromosomal position of interest that showed different cnv pattern

repeat discovery

There are two ways to find repeats in a genome:
1. find repeats directly from raw NGS reads (Repeat explorer - can be done in galaxy)
2. find repeats from the de novo assembled genome (Repeatmasker - can be done on galaxy or command line)

repeatmasker

To find repeats in your target genome with repeatmasker, we first need an assembled genome (repeatmasker do not accept NGS read directly).

So we will perform a de novo assembly from our NGS data.

de novo assembly

In this case, we will be using minia (http://minia.genouest.org/files/manual.pdf) to do the de novo assembly. Before doing de novo assembly, we need to determine the optimum k-mer (DNA subsequences length used for assembly). We will use kmergenie in this case.

k-mers: the length of the raw reads to be fragmented before performing the genome assembly

de novo assembly pipeline (run kmergenie, then minia)

1. list reads file one per line (kmergenie can only receive on read file or list of read files)

```
ls -1 *.fastq.gz > reads_files
```

2. run k-mergenie

```
'/home/user/Desktop/DDrive/kmergenie-1.7051/kmergenie' reads_files
```

3. assembly using minia with the optimum k-mer from k-mergenie

```
'/mnt/D/gatb-minia-pipeline/minia/minia'  -in reads_file -kmer-size 47 -out minia_47mer_hydrangea.fasta
```

4. After our de novo assembled genome is ready, we can then run repeatmasker.

```
'/mnt/D/RepeatMasker/RepeatMasker' -species hydrangea minia_47mer_hydrangea.contigs.fa
```

After repeatmasker is complete, it will return the repeat-masked genome, repeat families and location in genome. We are interested in the .tbl file that showed total number of repeat elements present in the genome according to repeat families.
Here is the example Arabidopsis repeats present in the hybrid genome (from the .tbl file).

number of length percentage elements occupied of sequence

Retroelements 11353 1675759 bp 0.06 %
SINEs: 90 9584 bp 0.00 %
Penelope 0 0 bp 0.00 %
LINEs: 1482 168513 bp 0.01 %
CRE/SLACS 0 0 bp 0.00 %

```
L2/CR1/Rex 0 0 bp 0.00 %
R1/LOA/Jockey 0 0 bp 0.00 %
R2/R4/NeSL 0 0 bp 0.00 %
RTE/Bov-B 0 0 bp 0.00 %
L1/CIN4 1468 166403 bp 0.01 %
LTR elements: 9781 1497662 bp 0.05 %
BEL/Pao 0 0 bp 0.00 %
Ty1/Copia 4143 382708 bp 0.01 %
Gypsy/DIRS1 5564 1109002 bp 0.04 %
Retroviral 0 0 bp 0.00 %

DNA transposons 14571 1377767 bp 0.05 %
hobo-Activator 1392 214452 bp 0.01 %
Tc1-IS630-Pogo 192 28079 bp 0.00 %
En-Spm 0 0 bp 0.00 %
MuDR-IS905 0 0 bp 0.00 %
PiggyBac 0 0 bp 0.00 %
Tourist/Harbinger 987 92057 bp 0.00 %
Other (Mirage, 0 0 bp 0.00 %
P-element, Transib)

Rolling-circles 0 0 bp 0.00 %

Unclassified: 2579 289629 bp 0.01 %

Total interspersed repeats: 3343155 bp 0.11 %

Small RNA: 1257 134738 bp 0.00 %

Satellites: 12761 3740017 bp 0.13 %
Simple repeats: 1495024 68632557 bp 2.35 %
Low complexity: 208577 10894574 bp 0.37 %
==================================================
```