PacBio_PCR_straglr_to_downstream_analysis_steps

sample: PacBio target PCR bam files on TERT and CLPTM1L region

analysis tools used: straglr, cutadpat, and bioawk

straglr tool for analyzing short-tandem repeat genotyping using long reads:

Github link: https://github.com/bcgsc/straglr

Install conda in your biowulf steps: https://hpc.nih.gov/apps/python.html#envs

go to desire folder, export TMPDIR

```
export TMPDIR=/lscratch/$SLURM_JOB_ID
```

download conda to your folder

```
wget https://github.com/conda-forge/miniforge/releases/latest/download/Mambaforge-Linux-x86_64.sh
```

install conda, -p is creating installed conda folder

```
bash Mambaforge-Linux-x86_64.sh -p /data/$USER/conda -b
```

remove installation app

```
rm Mambaforge-Linux-x86_64.sh
```

After sourcing the conda init file, activate the base environment and update the conda package manager which itself is just a package:

```
source /data/$USER/conda/etc/profile.d/conda.sh && source /data/$USER/conda/etc/profile.d/mamba.sh
```

to make things easier you can create a file called *myconda* in a directory on your path such as ~/bin. This could be done like so (assuming the same paths as we used here).

```
mkdir -p ~/bin

cat <<'__EOF__' > ~/bin/myconda

__conda_setup="$('/data/$USER/conda/bin/conda' 'shell.bash' 'hook' 2> /dev/null)"
if [ $? -eq 0 ]; then
    eval "$__conda_setup"
else
    if [ -f "/data/$USER/conda/etc/profile.d/conda.sh" ]; then
        . "/data/$USER/conda/etc/profile.d/conda.sh"
    else
        export PATH="/data/$USER/conda/bin:$PATH"
    fi
fi
unset __conda_setup

if [ -f "/data/$USER/conda/etc/profile.d/mamba.sh" ]; then
```

```
    . "/data/$USER/conda/etc/profile.d/mamba.sh"
fi
__EOF__
```

everytime you need to use conda, just type:

`source myconda`

once done, can use conda function

Tools required for analysis, can use `module load` to load tools in your biowulf environment, and activate conda `source myconda` after setup environment.

to set up an environment in biowulf https://hpc.nih.gov/docs/userguide.html

ex: set up 8 CPUs and 5 Gigabytes of memory in the norm (default) partition

`sinteractive --mem=5g --cpus-per-task=8`

samtools https://hpc.nih.gov/apps/samtools.html

Bioawk https://hpc.nih.gov/apps/bioawk.html

cutadapt https://hpc.nih.gov/apps/cutadapt.html

Histogram plot example for determine repeat size/copy number, thanks to Oscar's script

cutadapt and bioawk PCR amplicons approach

Extract reads id/sequecne from bam files go to folder of bam files, use samtools to get all the read sequence from bam file.

```
for i in *.bam
do
ff=‘echo $i | sed ’s/.bam//’‘
samtools view ${i} | awk ’{print ">"$1 "\t" $10}’ |sort | uniq | tr ’\t’ ’\n’ > ../fasta_files/${ff}_fas
done
```

---

using the TERT real PCR primers example

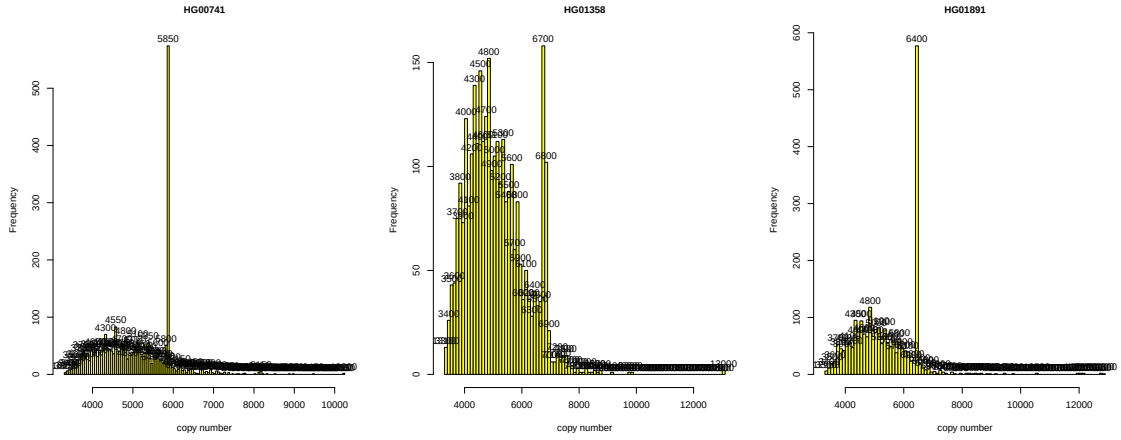
```
TERT_R4
F: CAGAAGGGAGGAAGCAGACA  ===>
R: ACCACGCCGAGTCAGATAAG  <===

TERT_R2
F: AGCAAGCCTCCAACTCGCAG ===>
R: TGTGCCGTGTGTGTCTTGCT <===
```

cutadapt to get the fasta files of sequence in PCR set, and use bioawk to count the length of sequence

```
# Need reverse complement for reverse primer when type primer
for i in *.txt
do
cutadapt --discard-untrimmed -g CAGAAGGGAGGAAGCAGACA ${i} 2> /dev/null | cutadapt --discard-untrimmed -a
done

# biowak

for i in set1_*
do
bioawk -c fastx '{ print $name, length($seq) }' < ${i} > cat_${i}
done

# if need 100% align, example below:

cutadapt --discard-untrimmed -g "CCTGGAAGCCCTGCCCACCGGCCAC;max_error_rate=0...GGTAGCTTCCGCTGCAGCGGGGATG
```

```
library(dplyr)
# library(xlsx)
# library(openxlsx)
library(reshape2)
library(tidyr)
library(ggplot2)

setwd('~/Desktop/markdown_for_straglr_steps/fasta_files/')

files_list = list.files(path = '.',pattern = 'cat_')

# ls_set.primer <- list()
for ( i in files_list ) {

  f.name <- gsub("cat_|_.*", "", i)
 # print(f.name)

  # tr.name <- gsub("*.*/cat_set|_.*", "", i)
  # tr.name
  tr.name = 'TERT_repeat'
```

```
  # primer.set <- gsub("*.*/filtered_by_|/.*", "", i)
  primer.set = 'TERT_targeted_primer'
  tryCatch({

    tmp <- read.table(i, header = FALSE, sep = "", fill = TRUE)
    tmp$file_name <- f.name
    tmp <- tmp[ complete.cases(tmp$V2), ]
    names(tmp) <- c("readID", "readLength_bp", "file_name")

    n_occur1 <- data.frame(table(tmp$readLength_bp))
    sum(n_occur1$Freq)

    tmp.unfiltered <- tmp
    if ( tr.name == "TERT_repeat" ) {
      tmp <- tmp[ which(tmp$readLength_bp >= 2500), ]
    } else { NULL }

    n_occur2 <- data.frame(table(tmp$readLength_bp))
    sum(n_occur2$Freq)
    h <- hist(tmp$readLength_bp, breaks = 100, main = f.name,plot = F)
    # hist(tmp$readLength_bp, breaks = nrow(n_occur2),labels = as.character(h$breaks),main = f.name)
    hist(tmp$readLength_bp,
         breaks = 100,
         labels = as.character(h$breaks),
         main = f.name,cex.main = 1,cex.lab = 1,col = "yellow",xlab = "copy number",# for C3
         )


  }, error=function(e){cat("ERROR :",conditionMessage(e), "\n")})

}
```



plotting

---

read/id size from bam files approach Go to folder where bam files are located and use script for extract reads and sequences from bam files and made into .txt files

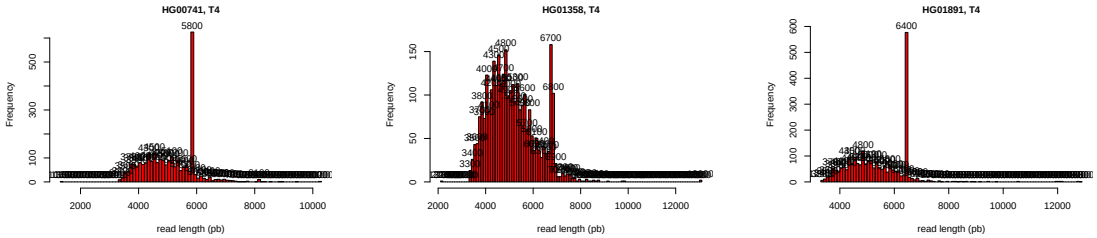
```
for i in *.bam
do
ff=‘echo $i | sed ’s/.bam//’‘
samtools view ${i} | awk ’{print ">"$1 "\t" $10}’ |sort | uniq | tr ’\t’ ’\n’ > ../fasta_files/${ff}_fas
done
```

Using Bioawk to get read length from sequnces.

```
for i in *.txt ; do bioawk -c fastx ’{ print $name, length($seq) }’ < ${i} > ${i}_read_len.txt; done
```

```
setwd('~/Desktop/markdown_for_straglr_steps/fasta_files/read_length/')
# plot histograms read sizes unfiltered

files_list <- list.files('.',pattern = '.txt')

for (i in files_list) {

  f.name <- gsub("_.*", "", i)
  tr.name <- gsub("^[A-Za-z0-9]+_|_.*", "", i)
  if ( tr.name == "TERTr4" ) {
    tr.name <- "T4"
  } else {
    tr.name <- "C3"
  }

  f <- read.table(file = i, header = FALSE, sep = "\t", stringsAsFactors = FALSE, fill = TRUE)
  names(f) <- c("read", "size")
  f$sampleID <- f.name
  n_rows.f <- nrow(f)

  ## for reads number
  tryCatch({

    f.set <- f

    h.plot <- hist(f.set$size,
                   breaks = 100, plot =F )
    par(mar=c(4,4,2,2))
    par(oma=c(1,1,1,1))
    hist(f.set$size,
         breaks = 100,
         labels = as.character(h.plot$breaks),
         main = paste0(f.name, ", ", tr.name),
         cex.main = 1,
         cex.lab = 1,
         col = "red",
         xlab = "read length (pb)",
         ylim = c(0,max(h.plot$counts)+20) # for T4
         #ylim = c(0,max(h.plot$counts)+10) # for C3
    )
```

```
  }, error=function(e){cat("ERROR :",conditionMessage(e), "\n")})

}
```



plotting script

---

straglr tsv output approach Install straglr using conda in Biowulf/own computer

`git clone https://github.com/bcgsc/straglr.git`

go to straglr folder

`cd straglr`

use conda to install straglr

`conda env create --name straglr --file=environment.yaml`

straggler problem in version 1.3.0, install 1.2.0 instead

`conda activate straglr`

`conda install straglr=1.2.0`

Check version

`straglr.py --version`

to see if it can execute, type

`./straglr.py`

How to scoring repeates using straglr

set working dict to the folder where files are located

```
setwd('~/Desktop/markdown_for_straglr_steps/')
```

make .bed file of repeat region and download hg38.fa from biowulf/ucsc. . . etc

The bed file (tab seperate) look like this:

```
cat TERT_R4_hg38.bed
```

```
## chr5 1288761 1291834 TERT_R4_GGACACCCGGGGACCGCGCCTCACTCACCCTGCACGTGACAG
```

Run the sample using HG01433_pacbio.sorted.bam as example: ###### bascic straglr function: straglr.py bam genome_fasta out_prefix

```
straglr.py HG01433_pacbio.sorted.bam hg38.fa ./output --loci clp_R3_hg38.bed
```

Can make a linux script to loop all the bam files

```
for i in *.bam
do
ff=‘echo $i | sed ’s/.bam//’‘
straglr.py ${i} ~/Desktop/straglr_scoring_tool_for_long_read/toolinpus_and_HPRC_bams/hg38.fa ~/Desktop/s
 --max_str_len 100 #optional
echo "done ${ff}"
done
```

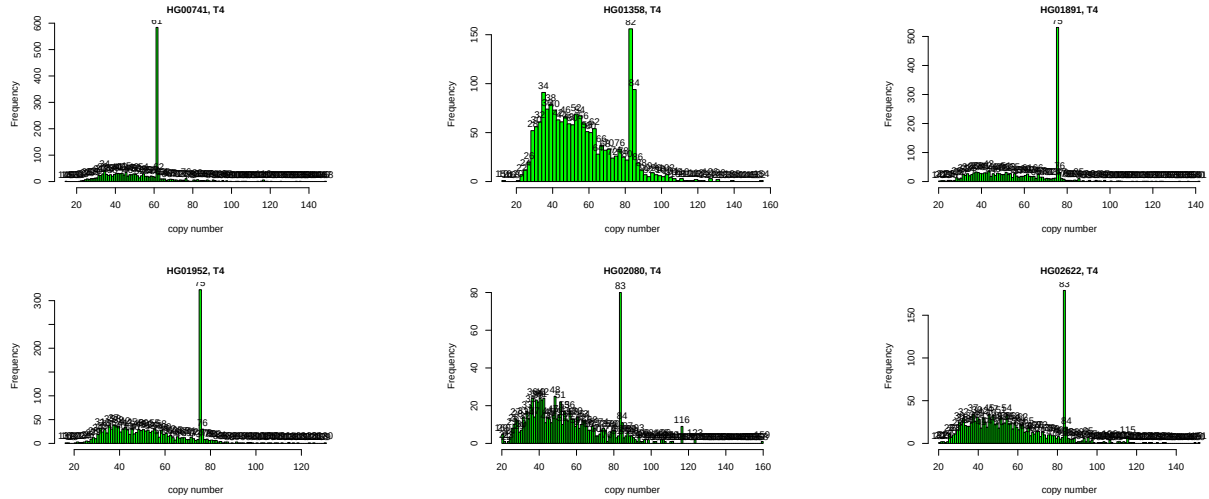
The output will generate .tsv and .bed file, tsv has detailed information

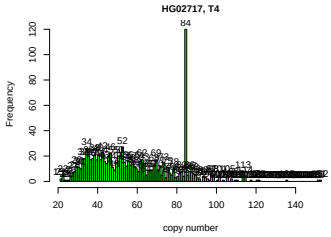
```
##   X.chrom   start     end                          repeat_unit          genotype
## 1    chr5 1332959 1333821 GGGACTACTGTATACACACCGGATGAGGATAAGGG 190.2(17);93.3(3)
## 2    chr5 1332959 1333821 GGGACTACTGTATACACACCGGATGAGGATAAGGG 190.2(17);93.3(3)
## 3    chr5 1332959 1333821 GGGACTACTGTATACACACCGGATGAGGATAAGGG 190.2(17);93.3(3)
## 4    chr5 1332959 1333821 GGGACTACTGTATACACACCGGATGAGGATAAGGG 190.2(17);93.3(3)
## 5    chr5 1332959 1333821 GGGACTACTGTATACACACCGGATGAGGATAAGGG 190.2(17);93.3(3)
## 6    chr5 1332959 1333821 GGGACTACTGTATACACACCGGATGAGGATAAGGG 190.2(17);93.3(3)
##                  read copy_number size read_start allele
## 1 SRR18189642.2507077       211.3 7395      14903  190.2
## 2 SRR18189642.4101788       210.2 7356       9902  190.2
## 3 SRR18189642.4359315       210.2 7357       4257  190.2
## 4 SRR18189642.1029907       210.1 7353       9466  190.2
## 5 SRR18189642.4719843       210.0 7351       8268  190.2
## 6  SRR18189642.994697       209.8 7343       3973  190.2
```

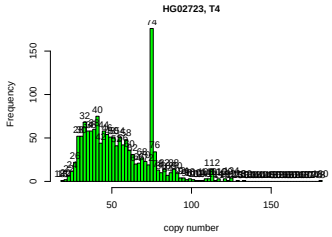
plotting

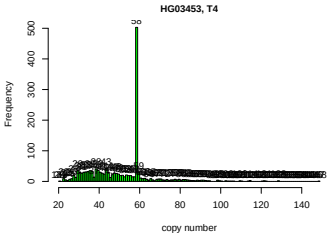
```
# loop through the tsv files
setwd('~/Desktop/markdown_for_straglr_steps/straglr_result_TERT_HG/')
files_list <- list.files('.',pattern = ".tsv")
# ls_out.straglr.all <- list()


for (i in files_list) {
  f.name <- gsub("_.*", "", i)
  tr.name <- gsub("^[A-Za-z0-9]+_|_.*", "", i)
  if ( tr.name == "TERTr4" ) {
    tr.name <- "T4"
  } else {
    tr.name <- "C3"
  }
  f <- read.table(file = i, header = FALSE, sep = "\t", stringsAsFactors = FALSE)
  # this colname is for straglr version 1.2, the v1.3.0 will have 13 column with "strand" between "size
  names(f) <- c("chrom", "start", "end", "repeat_unit", "genotype",
                "read", "copy_number", "size", "read_start", "allele")
  f[ f == "-" ] <- NA
  f$sampleID <- f.name
```

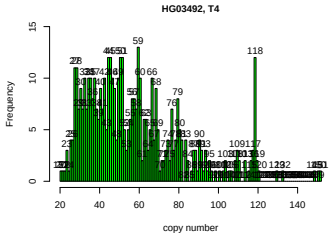
```
  n_rows.f <- nrow(f)
  # ls_out.straglr.all[[tr.name]][[f.name]] <- f

  # basic stats 1
  getmode <- function(v) {
    uniqv <- unique(v)
    uniqv[which.max(tabulate(match(v, uniqv)))]
  }
  n_mean <- mean(f$copy_number)
  n_mode <- getmode(f$copy_number)
  n_occur1 <- data.frame(table(f$copy_number))
  names(n_occur1) <- c("allele", "nReads")

  h.plot <- hist(f$copy_number,breaks = 100, plot = F )
  # making plot into single figure output
  par(mar=c(4,4,2,2))
  par(oma=c(1,1,1,1))

  hist(f$copy_number,
         # breaks = length(unique(f.set$copy_number)),
         breaks = 100,
         labels = as.character(h.plot$breaks),
         main = paste0(f.name, ", ", tr.name),
         cex.main = 1,
         cex.lab = 1,
         col = "green",
         xlab = "copy number",
         # ylim = c(0,max(h.plot$counts)+10) # for T4
         ylim = c(0,max(h.plot$counts)+2) # for C3
         )


   ####
}
```

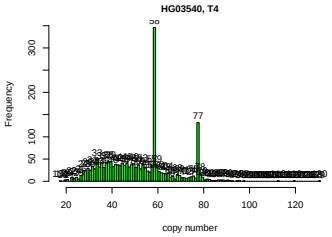


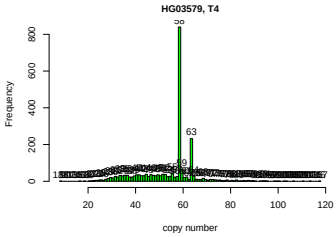

HG02717, T4
Frequency
copy number


HG02723, T4
Frequency
copy number


HG03453, T4
Frequency
copy number


HG03492, T4
Frequency
copy number


HG03540, T4
Frequency
copy number


HG03579, T4
Frequency
copy number