

----- README.txt -----

Supplementary Code S1

Publication

Decoding common and complex FSHD genotypes using methylation profiling and long-read sequencing
Scharf, Lucas et al., Erdmann (last author)
Brain Communications, 2026

Contents

This archive contains the custom scripts used for read sampling and statistical analysis as described in the Methods section of the above manuscript. The code is provided for transparency and reproducibility purposes. File paths and sample-specific parameters require adjustment before use.

File	Description
fshd_downsampling.sh	Main script for data evaluation; loops through the downsampling steps and invokes subsample_reads.py, calculateMethPerRU.sh and calculateSlope.R
subsample_reads.py	Random sampling without replacement for downsampling replicates
calculateMethPerRU.sh	Calculates average methylation values per repeat unit
calculateSlope.R	Calculates the slope of the methylation values per repeat unit
calculateDownsamplingStats.R	Statistical analysis of downsampling experiments (IQR, exponential fit)

Dependencies

- Python (v3.9): Python Software Foundation. Available at <http://www.python.org>
- R (v4.0.0): R Core Team. R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. 2020. Available at <https://www.R-project.org/>
- bash
- modbam2bed (v0.9.5): Oxford Nanopore Technologies. Available at <https://github.com/epi2me-labs/modbam2bed>
- samtools (v1.14): Danecek P, Bonfield JK, Liddle J, et al. Twelve years of SAMtools and BCFtools. GigaScience. 2021;10(2):giab008. doi:10.1093/gigascience/giab008

Usage Notes

- File paths require adjustment to match the local directory structure
- This code has not been optimized for general use and may require adaptation for other datasets
- For questions regarding the code, please contact florentine.scharf@mgz-muenchen.de

----- fshd_downsampling.sh -----

```
#!/bin/bash
```

```
### BAM file containg only reads suitable for downsampling, i.e. unambiguously assigned to one allele
### mapped against T2T anchored to the distal end
input_bam=$1
### tsv file containing read ID, number of repeat units and information on whether the read is full length (y) or not (n)
read_characterization=$2
```

```
samtools view "$input_bam" | awk 'BEGIN {FS="\t"} {print $1}' | sort -u > all_reads.txt
```

```
### iterate through different downsampling levels, each 30 times
```

```
IFS=$'\n'
```

```
seed=0
```

```
for reads in 1 2 3 4 5 6 7 8 9 10 15 20
```

```
do
```

```
    for rep in $(seq 1 30)
```

```
    do
```

```
        mkdir "$seed"."$reads"
```

```
        python3 subsample_reads.py all_reads.txt "$reads" "$seed" "$seed"."$reads"/downsampled_reads.txt
```

```
        for line in $(cat "$seed"."$reads"/downsampled_reads.txt)
```

```
        do
```

```
            grep "$line" "$read_characterization" >> "$seed"."$reads"/"$read_characterization"_downsampled
```

```
        done
```

```
        samtools view -N "$seed"."$reads"/downsampled_reads.txt -o "$seed"."$reads"/"$input_bam".subset.bam
```

```
"$input_bam"
```

```
        samtools sort "$seed"."$reads"/"$input_bam".subset.bam > "$seed"."$reads"/"$input_bam".subset.sorted.bam
```

```
        samtools index "$seed"."$reads"/"$input_bam".subset.sorted.bam
```

```
        ./calculateMethPerRU.sh "$seed"."$reads"/"$input_bam".subset.sorted.bam
```

```
"$seed"."$reads"/"$input_bam".slope.bedmethyl
```

```
        distal=$(tail -n 1 "$seed"."$reads"/"$input_bam".slope.bedmethyl | cut -d'\t' -f 4)
```

```
        repeat=$(awk 'BEGIN {FS="\t"} {if($4>0) {print $3}}' "$seed"."$reads"/"$input_bam".slope.bedmethyl | head -n 1)
```

```
        slope=$(Rscript ./calculateSlope.R "$seed"."$reads"/"$input_bam".slope.bedmethyl "$repeat" | sed -n '2p')
```

```
        echo -e "$reads\t$distal\t$slope" >> output.tsv
```

```
        seed=$((seed+1))
```

```
    done
```

```
done
```

```
----- subsample_reads.py -----
```

```
import random
```

```
import sys
```

```
input_file = sys.argv[1]
```

```
n_reads = int(sys.argv[2])
```

```
seed = int(sys.argv[3])
```

```
output_file = sys.argv[4]
```

```
with open(input_file) as f:
```

```
    reads = [line.strip() for line in f if line.strip()]
```

```
if n_reads > len(reads):
```

```
    print(f"WARNING: Only {len(reads)} reads verfügbar, less than the required {n_reads}")
```

```
    sys.exit(1)
```

```
random.seed(seed)
```

```
sampled = random.sample(reads, n_reads)
```

```
with open(output_file, "w") as f:
```

```
    f.write("\n".join(sampled) + "\n")
```

```
----- calculateMethPerRU.sh -----
```

```
#!/bin/bash
```

```

bam=$1
out_bam=$2
### Array to define repeat unit borders based on KpnI positions in T2T
declare -a ru_arr=("193434262" "193437559" "193440855" "193444152" "193447448" "193450744" "193454040"
"193457345" "193460639" "193463937" "193467233" "193470536" "193473832" "193477127" "193480424" "193483717"
"193487016" "193490314" "193493614" "193496914" "193500211" "193503509" "193506805" "193510102" "193513399"
"193516697" "193519995" "193523291" "193526588" "193529884" "193533181" "193536477" "193539783" "193543078")

start=-
ru=33
ruNeg=-32
for i in "${ru_arr[@]}"
do
    if [[ "$start" == "-" ]]
    then
        start=$((i+1))
    else
        modbam2bed --aggregate -e -m 5mC --cpg -r chr4:"$start"-"$i" t2t.fa "$bam" > ru.bedmethyl
        mean=$(awk 'BEGIN {FS="\t"} {if ($5>0) print $0}' ru.bedmethyl | awk '{ sum += $11; n++ } END { if (n > 0) print sum
/ n; }')
        echo -e "chr4:$start-$i\t$ru\t$ruNeg\t$mean" >> "$out_bam"
        start=$((i+1))
        ru=$((ru-1))
        ruNeg=$((ruNeg+1))
    fi
done

```

----- calculateSlope.R -----

```

args <- commandArgs(trailingOnly = TRUE)

if (length(args) < 2) {
    stop("Usage: Rscript script.R <input_file> <var1> <var2>", call. = FALSE)
}

input_file <- args[1]
ru <- args[2]

if (!file.exists(input_file)) {
    stop(paste("File not found:", input_file), call. = FALSE)
}

data <- read.table(input_file, sep="\t", header=F)
colnames(data) <- c("pos", "RUpos", "RU", "methylation")

rus=as.numeric(ru)
data<-data[data$RU>rus,]
rusAbs=abs(rus)
rusPos=seq(1,rusAbs)
data$RU<-rusPos
df<-data
df.lm <- lm(methylation~RU, df)

intercept<-coef(df.lm)[[1]]
slope<-coef(df.lm)[[2]]

cat(intercept, "\n")
cat(slope, "\n")

```

----- calculateDownsamplingStats.R -----

```

#####
### Supplementary Code: Read depth threshold determination ###
#####

```

```

#### Step 1: Exponential function fit to median IQR across samples for manual inspection
#### Step 2: Proportion of replicates exceeding the threshold for pathogenic alleles (Mean + 2SD of contracted alleles = 48.1%)
#### The minimum read threshold is defined as the first read depth at which this proportion remains persistently below 10% for all higher read depths

```

```

#### Input:
#### One .tsv per sample, columns: Full_Length, Slope, no header
#### The output of fshd_downsampling.sh can be used as input here
#### Paths will have to be adjusted accordingly

```

```

#### Output:
#### IQR table printed to console
#### Exponential fit parameters printed to console

```

```

files <- list(
  "A1" = "A1_input.tsv",
  "A2" = "A2_input.tsv",
  "A3" = "A3_input.tsv",
  "A4" = "A4_input.tsv",
  "A5" = "A5_input.tsv",
  "A7" = "A7_input.tsv",
  "A11" = "A11_input.tsv",
  "A12" = "A12_input.tsv",
  "C1" = "C1_input.tsv"
)

#### Read and combine data
read_sample <- function(path, sample_name) {
  df <- read.delim(path, sep = "\t", header = FALSE,
    stringsAsFactors = FALSE,
    col.names = c("Full_Length", "Dist_Meth", "Slope"))
  df$Full_Length <- as.integer(df$Full_Length)
  df$Dist_Meth <- as.numeric(df$Dist_Meth)
  df$Slope <- as.numeric(df$Slope)
  df <- df[!is.na(df$Full_Length) & !is.na(df$Dist_Meth) &
    !is.na(df$Slope) & df$Full_Length > 0, ]
  df$Sample <- sample_name
  df
}

df_list <- mapply(read_sample, files, names(files), SIMPLIFY = FALSE)
df <- do.call(rbind, df_list)
df$Sample <- factor(df$Sample, levels = names(files))

```

```

#### Downsampling levels tested
fl_levels <- c(1:10, 15, 20)

```

```

#####
### MEDIAN IQR PER DOWNSAMPLING LEVEL ###
#####

```

```

#### For each downsampling level and each sample, compute IQR across replicates, then take the median IQR across all samples as a robust summary.

```

```

iqr_df <- do.call(rbind, lapply(fl_levels, function(level) {
  iqr_meth <- sapply(levels(df$Sample), function(s) {
    vals <- df$Dist_Meth[df$Full_Length == level & df$Sample == s]
    if (length(vals) >= 3) IQR(vals, na.rm = TRUE) else NA
  })
  iqr_slope <- sapply(levels(df$Sample), function(s) {
    vals <- df$Slope[df$Full_Length == level & df$Sample == s]
    if (length(vals) >= 3) IQR(vals, na.rm = TRUE) else NA
  })
}))

```

```

}))
data.frame(
  Full_Length = level,
  iqr_meth = median(iqrs_meth, na.rm = TRUE),
  iqr_slope = median(iqrs_slope, na.rm = TRUE)
)
}))
iqr_df <- iqr_df[!is.na(iqr_df$iqr_meth), ]

cat("=== Median IQR per downsampling level ===\n")
print(iqr_df)

#####
### EXPONENTIAL FIT ###
#####

### Model:  $IQR(FL) = a * \exp(-b * FL) + c$ 
### a = amplitude (total decay from maximum to plateau)
### b = decay rate
### c = asymptote (minimum achievable dispersion)
### The 90% decay cutoff is provided for reference but is NOT used as the primary threshold criterion.

fit_exp_decay <- function(fl_vals, iqr_vals, label) {
  fit <- nls(
    iqr_vals ~ a * exp(-b * fl_vals) + c,
    start = list(a = max(iqr_vals) - min(iqr_vals),
                 b = 0.3,
                 c = min(iqr_vals)),
    control = nls.control(maxiter = 200)
  )
  a <- coef(fit)["a"]
  b <- coef(fit)["b"]
  c <- coef(fit)["c"]

  ### decay achieved at downsampling level 6 (established threshold)
  pct_at_fl6 <- (1 - exp(-b * 6)) * 100

  cat(sprintf("\n--- %s ---\n", label))
  cat(sprintf("a = %.4f, b = %.4f, asym = %.4f\n", a, b, c))
  cat(sprintf("%% decay at downmapling 6: %.1f%%\n", pct_at_fl6))

  invisible(list(fit = fit, a = a, b = b, c = c,
                 pct_at_fl6 = as.numeric(pct_at_fl6)))
}

fit_meth <- fit_exp_decay(iqr_df$Full_Length, iqr_df$iqr_meth, "Distal methylation")
fit_slope <- fit_exp_decay(iqr_df$Full_Length, iqr_df$iqr_slope, "Slope")

```