# Basic usage of MCPtaggR

**August 9, 2023**

**Package**

MCPtaggR 0.99.1

## Contents

# 1 Introduction

MCPtaggR provides the functions to conduct mappable-collinear-polymorphic tag genotyping (MCPtagg). MCPtagg is a pipeline for next generation sequencing (NGS)-based genotyping technique to precisely detect SNPs and count reads on validated tags (short genome segments or regions) based on genome comparison information. Short NGS reads can not always be mapped to the proper positions on the reference genome if the reads were originally derived from a genome that have plenty polymorphisms and structural variations against the reference genome, e.g. mapping reads of a hybrid population that were derived from a cross between distant relatives. Polymorphic reads that were derived from a non-reference genome could result in mismapping and mapping bias. The mismapping and mappability bias lead biased observation of mapped reads on SNP markers where reads of either allele are preferentially mapped. To eliminate unexpected detection of such error-prone markers, MCPtagg count reads only mapped on MCP tags that are the genomic sequences/regions on which NGS reads obtained by RRS can be mapped uniquely and properly to call genotypes at SNPs. The list of MCP tags is obtained by whole genome sequence comparison between the genome sequences of the prarents of a given mapping/breeding population followed by sequencing read mappability assessment using simulated reads. The MCPtagg pipeline efficiently eliminates error-prone markers from your resultant genotype data obtained via a NGS-based genotyping technique, compared to the result produced by the conventional pipeline that maps reads on one reference genome. MCPtaggR currently supports NGS reads generated by reduced-representation sequence, e.g. genotyping-by-sequencing (GBS) and Restriction site associated DNA markers sequencing (RAD-seq), for a biparental population derived from a cross between inbred parents.

# 2 Prerequisites

`MCPtaggR` requires external tools: MUMmer and Subreads. Please install these two tools on your system and make sure their executable files can be executed from your R environment. For installation of MUMmer, please visit MUMmer's web site. For installation of Subread, please see the vignette of Rsubread.

# 3 Installation

You can install `MCPtaggR` from the GitHub repository.

```
if (!requireNamespace("devtools", quietly = TRUE))
    install.packages("devtools")

devtools::install_github("tomoyukif/MCPtaggR", build_vignettes = TRUE)
```

# 4 The basic pipeline

Load the package.

```
library("MCPtaggR")
```

Set paths to input files.

```
fq_dir <- system.file("extdata/fastq", package = "MCPtaggR")
ref_fn <- system.file("extdata/ref/IRGSP-1.0_genome.masked.fa.gz", package = "MCPtaggR")
alt_fn <- system.file("extdata/ref/ol_genome_chr.masked.fa.gz", package = "MCPtaggR")
mummer_dir <- system.file("extdata/MUMmer3.23", package = "MCPtaggR")
out_dir <- tempdir()
```

Run MUMmer.

```
out_fn <- "mummer_out"
R.utils::gunzip(ref_fn, overwrite = TRUE, remove = FALSE)
R.utils::gunzip(alt_fn, overwrite = TRUE, remove = FALSE)
ref_fn <- sub("\\.gz", "", ref_fn)
alt_fn <- sub("\\.gz", "", alt_fn)
mummer_fn <- run_mummer(ref_fn = ref_fn, alt_fn = alt_fn,
                        mummer_dir = mummer_dir,
                        out_dir = out_dir, out_fn = out_fn)
```

You can use pre-calculated mummer outputs from installation directory of MCPtaggR.

```
mummer_fn <- c(system.file("extdata/mummer/ol2nb_mcptagg.snps.gz", package = "MCPtaggR"),
               system.file("extdata/mummer/ol2nb_mcptagg.coord.gz", package = "MCPtaggR"))
```

Import and organize information of SNPs in collinear blocks.

```
snps <- mummer2SNPs(mummer_fn = mummer_fn)
```

Execute in-silico digestion of the given genomes.

```
read_len <- 150
re_names <- c("PstI", "MspI")
re <- makeRE(re_names)
fragment_len = c(0, 1000)
out_fn <- "dgout"
dg <- digestGenome(snps = snps, ref_fn = ref_fn, alt_fn = alt_fn,
                   out_dir = out_dir, read_len = read_len,
                   re = re, fragment_len = fragment_len)
```

Align the simulated digested fragments to the given genomes.

```
ar <- alignTAG(dg = dg, out_dir = out_dir,
               n_threads = 5, indexing = TRUE, reuse = TRUE)
```

Evaluate mappability of tags.

```
mcptag <- findMCPtag(ar = ar, dg = dg)
```

Count reads on the MCPtags.

```
fq_list <- findFASTQ(in_dir = fq_dir)
mcptagg <- mcptagg(mcptag = mcptag, fq_list = fq_list, out_dir = out_dir,
                   n_threads = 5, n_parallel = 3)
```

## 5 Post genotype calling workflow

The output GDS file can be manipulated using the GBScleanR package that provides functions for data visualization, filtering, genotyping error correction, and allele dosage estimation.

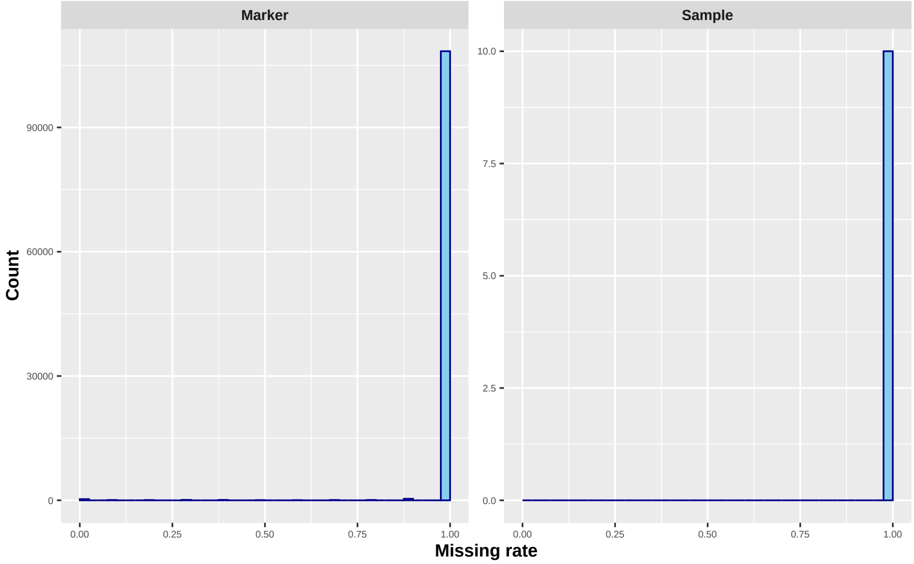
```
library(GBScleanR)
```

Open the connection to the GDS file.

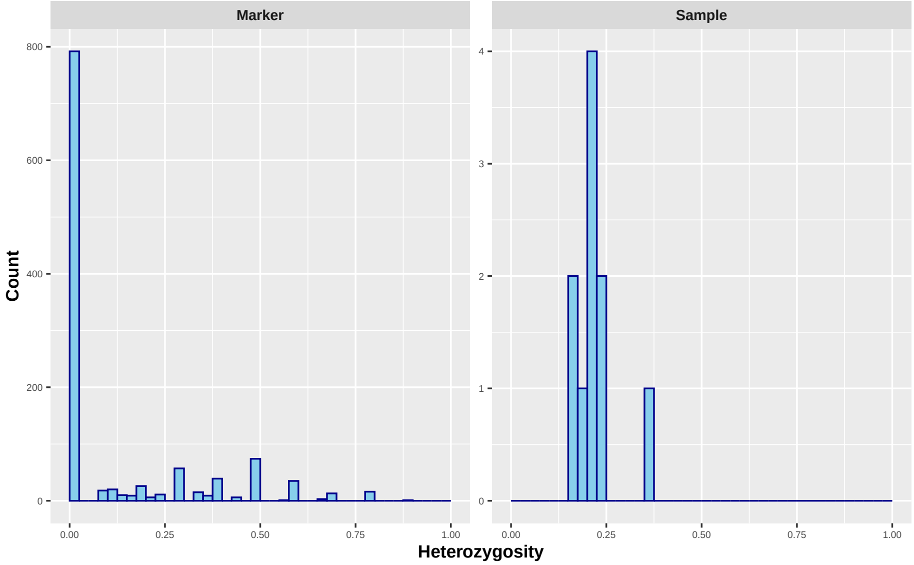
```
mcptagg <- system.file("extdata/genotype.gds", package = "MCPtaggR")
gds <- loadGDS(mcptagg)
```

Calculate and visualize summary statistics.

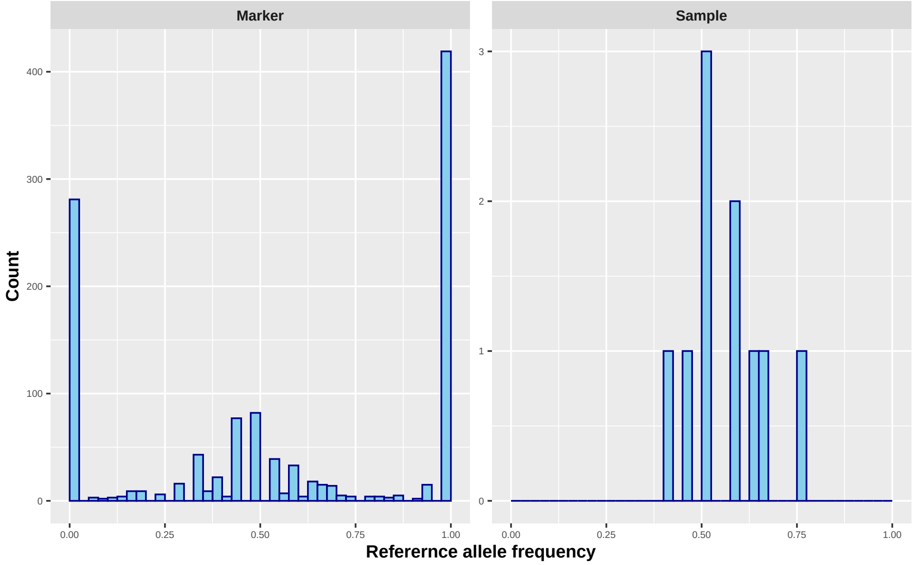
```
gds <- countGenotype(gds)
gds <- countRead(gds)
histGBSR(x = gds, stats = "missing")
```



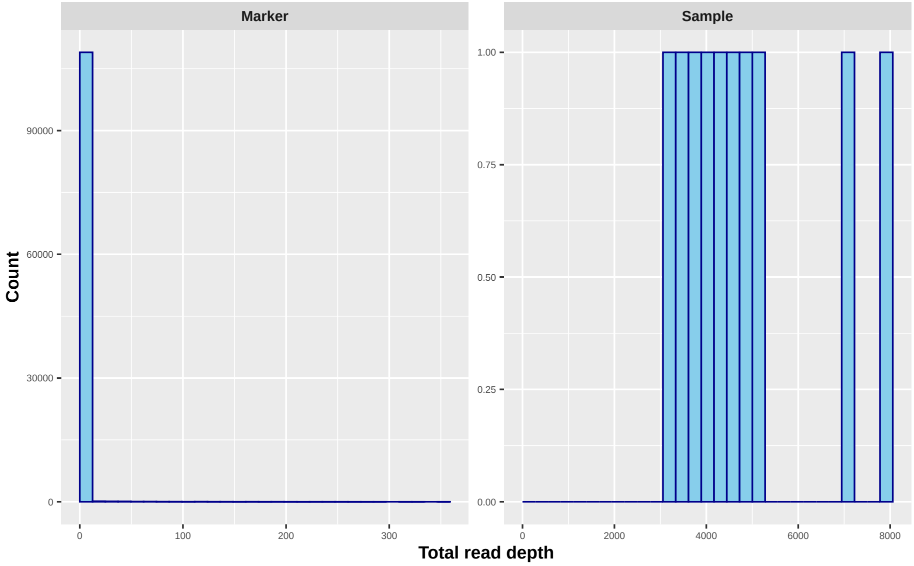
```
histGBSR(x = gds, stats = "het")
```



```
histGBSR(x = gds, stats = "raf")
```



```
histGBSR(x = gds, stats = "dp")
```

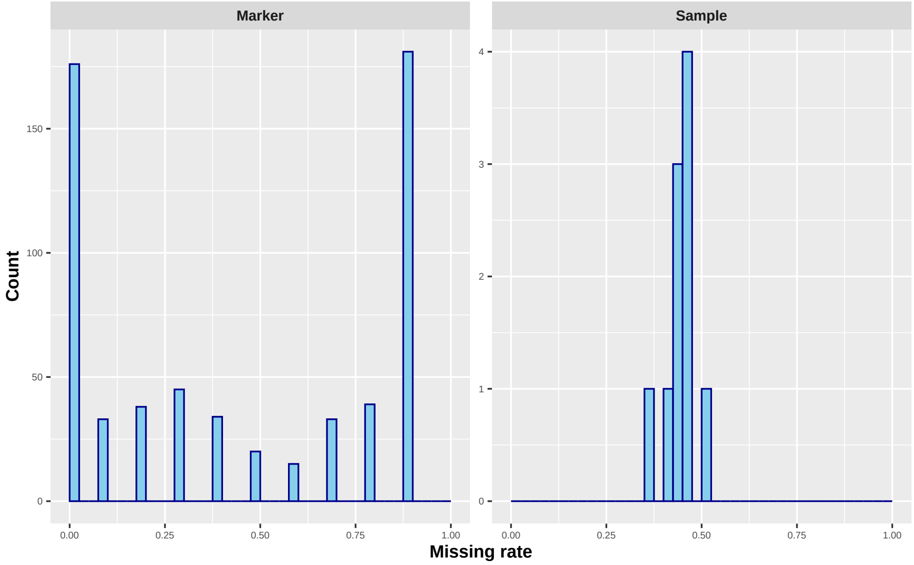


Apply filtering on SNPs.

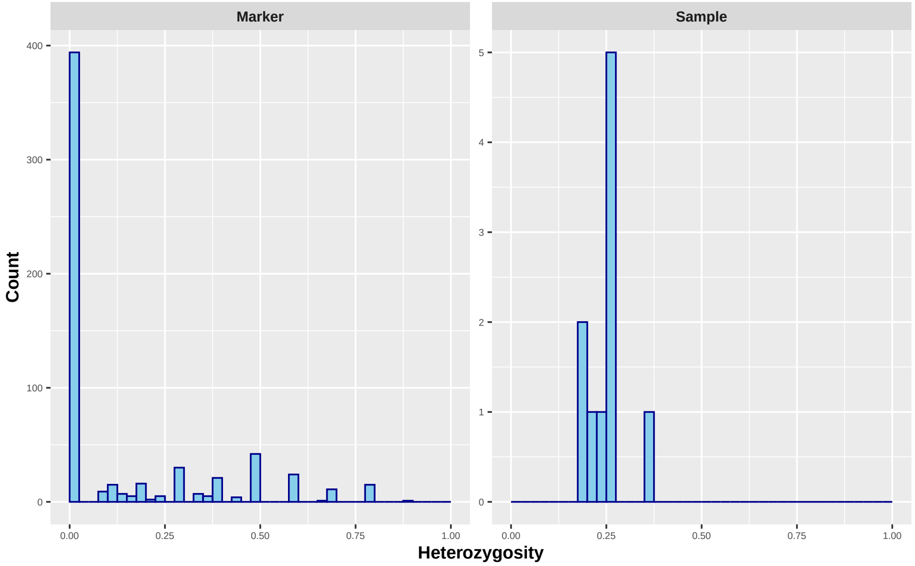
```
# Retain only one SNP if multiple SNPs in a 150 bp stretch.
gds <- thinMarker(object = gds, range = 150)
# Filter out SNPs at which all samples shows missing.
gds <- setMarFilter(object = gds, missing = 0.9999)
nmar(gds)
## [1] 614
```

Re-calculate and visualize summary statistics for the filtered data.

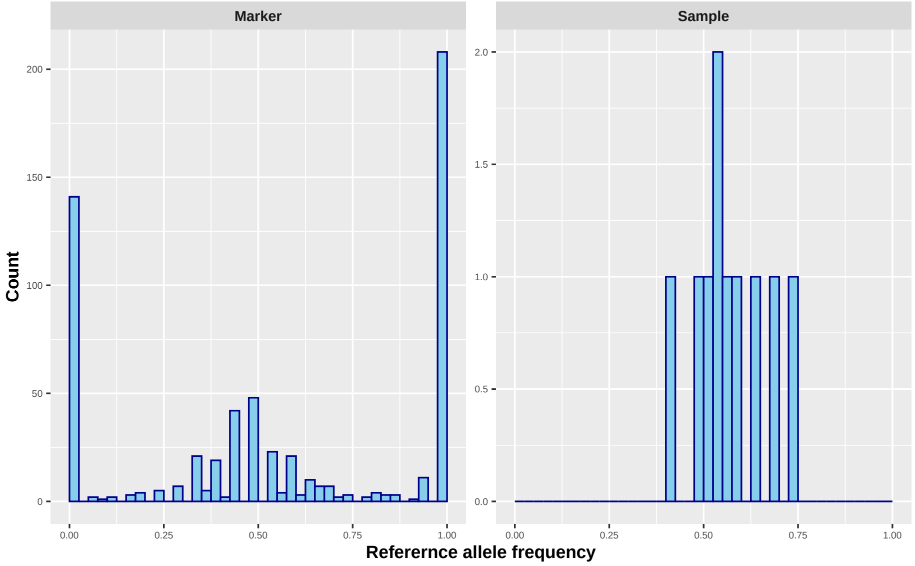
```
gds <- countGenotype(gds)
gds <- countRead(gds)
histGBSR(x = gds, stats = "missing")
```



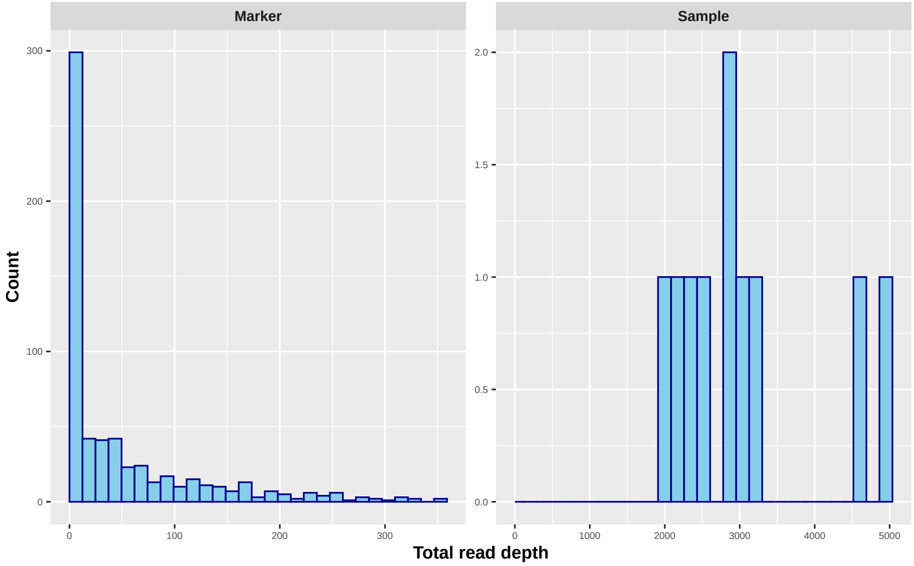
```
histGBSR(x = gds, stats = "het")
```



```
histGBSR(x = gds, stats = "raf")
```



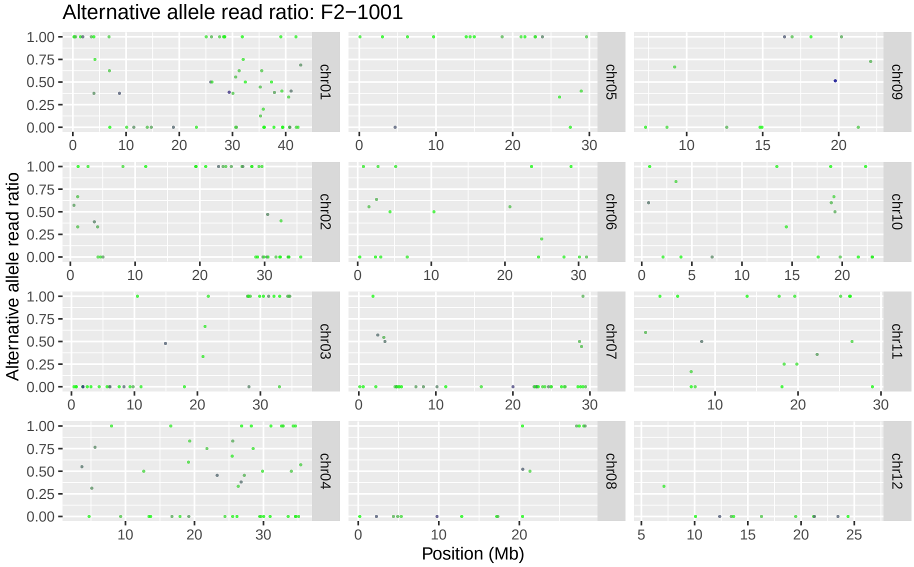
```
histGBSR(x = gds, stats = "dp")
```



Genotyping error correction and dosage estimation.

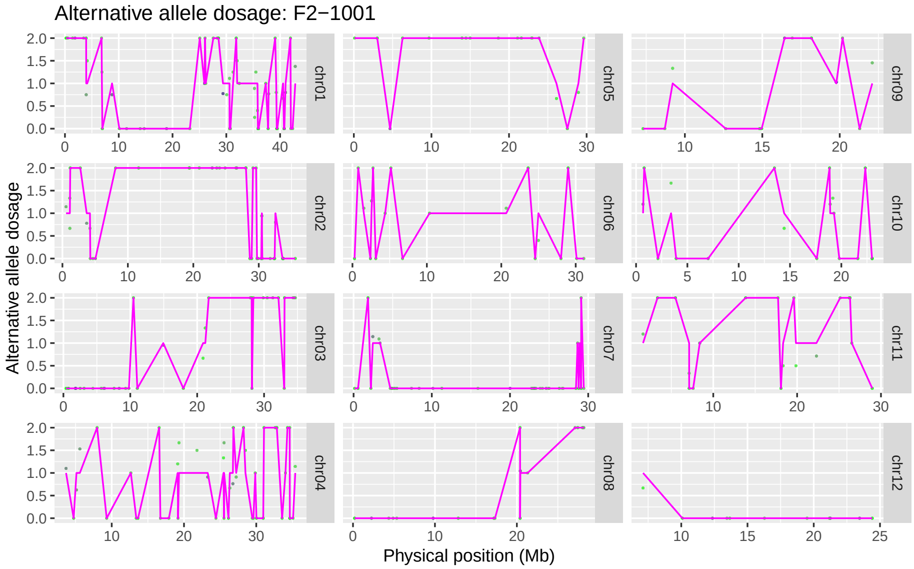
Visualize read ratio and estimated dosage

```
plotReadRatio(x = gds, ind = 1, alpha = 0.8)
```

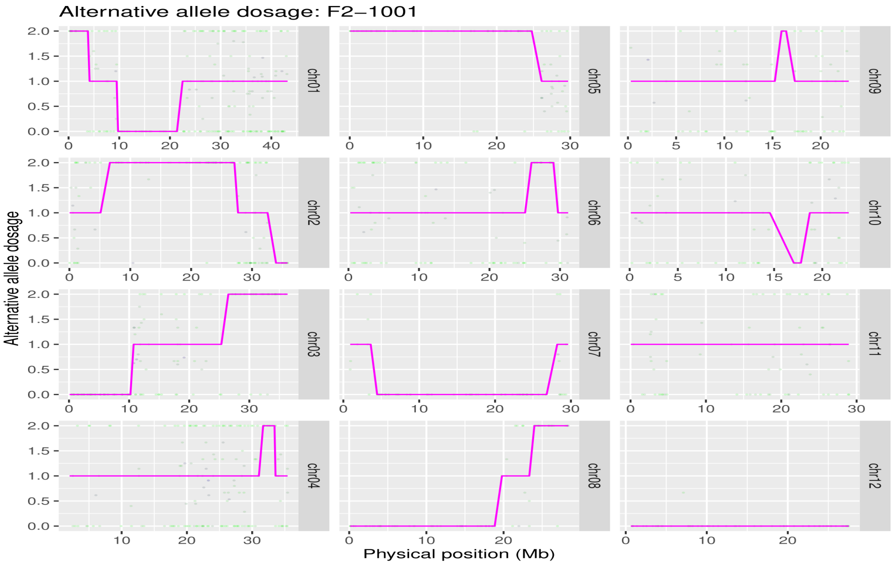


Due to the dosage estimation conducted on the data with too few reads and few samples, the estimated dosage is highly fluctuated and obviously contains error.

```
plotDosage(x = gds, ind = 1, alpha = 0.8)
```



The follwing plot shows the estimated dosage that generated by running estGen() on 813 samples.



# Session info

```
sessionInfo()
## R version 4.3.1 (2023-06-16)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 22.04.2 LTS
##
## Matrix products: default
## BLAS:   /usr/lib/x86_64-linux-gnu/openblas-pthread/libblas.so.3
## LAPACK: /usr/lib/x86_64-linux-gnu/openblas-pthread/libopenblasp-r0.3.20.so;  LAPACK version 3.10.0
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=en_US.UTF-8        LC_COLLATE=en_US.UTF-8
##  [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
##  [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
##
## time zone: Asia/Tokyo
## tzcode source: system (glibc)
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] GBScleanR_1.5.7  SeqArray_1.40.1  gdsfmt_1.36.1    MCPtaggR_0.99.1
## [5] BiocStyle_2.28.0
##
## loaded via a namespace (and not attached):
##  [1] tidyselect_1.2.0            farver_2.1.1
##  [3] dplyr_1.1.2                 blob_1.2.4
##  [5] Biostrings_2.68.1           bitops_1.0-7
##  [7] fastmap_1.1.1               RCurl_1.98-1.12
##  [9] GenomicAlignments_1.36.0    XML_3.99-0.14
## [11] digest_0.6.33               lifecycle_1.0.3
## [13] RSQLite_2.3.1               magrittr_2.0.3
## [15] compiler_4.3.1              rlang_1.1.1
## [17] tools_4.3.1                 utf8_1.2.3
## [19] yaml_2.3.7                  SNPRelate_1.34.1
## [21] data.table_1.14.8           rtracklayer_1.60.0
## [23] knitr_1.43                  labeling_0.4.2
## [25] S4Arrays_1.0.5              bit_4.0.5
## [27] DelayedArray_0.26.7         abind_1.4-5
## [29] BiocParallel_1.34.2         expm_0.999-7
## [31] withr_2.5.0                 purrr_1.0.1
## [33] BiocGenerics_0.46.0         grid_4.3.1
## [35] stats4_4.3.1                fansi_1.0.4
## [37] colorspace_2.1-0            ggplot2_3.4.2
## [39] scales_1.2.1                Rsubread_2.14.2
## [41] SummarizedExperiment_1.30.2 cli_3.6.1
## [43] rmarkdown_2.23              crayon_1.5.2
## [45] generics_0.1.3              RcppParallel_5.1.7
## [47] rstudioapi_0.15.0           rjson_0.2.21
## [49] DBI_1.1.3                   cachem_1.0.8
## [51] zlibbioc_1.46.0             parallel_4.3.1
## [53] BiocManager_1.30.21.1       XVector_0.40.0
## [55] restfulr_0.0.15             matrixStats_1.0.0
## [57] vctrs_0.6.3                 Matrix_1.6-0
## [59] bookdown_0.34               IRanges_2.34.1
## [61] S4Vectors_0.38.1            bit64_4.0.5
## [63] tidyr_1.3.0                 glue_1.6.2
## [65] codetools_0.2-19            gtable_0.3.3
## [67] GenomeInfoDb_1.36.1         BiocIO_1.10.0
## [69] GenomicRanges_1.52.0        munsell_0.5.0
## [71] tibble_3.2.1                pillar_1.9.0
## [73] htmltools_0.5.5             GenomeInfoDbData_1.2.10
## [75] BSgenome_1.68.0             R6_2.5.1
## [77] evaluate_0.21               lattice_0.21-8
## [79] Biobase_2.60.0              png_0.1-8
## [81] Rsamtools_2.16.0            memoise_2.0.1
## [83] DECIPHER_2.28.0             Rcpp_1.0.11
## [85] xfun_0.39                   MatrixGenerics_1.12.3
## [87] pkgconfig_2.0.3
```