Example Analysis using BiocMAP Output Objects

Nicholas J. Eagles[*1] ***and Leonardo Collado-Torres***[†1,2]

[1]Lieber Institute for Brain Development, Johns Hopkins Medical Campus
[2]Center for Computational Biology, Johns Hopkins University

[*]`nickeagles77@gmail.com` [†]`lcolladotor@gmail.com`

18 February 2022

Contents

1 Add Experiment Metadata to BiocMAP Outputs

The `bsseq` output objects from BiocMAP contain methylation and coverage info for our samples in the dataset. However, we're interested in exploring how this information relates back to sample metadata and phenotype information, present in an external file. Our first step will therefore be to load the BiocMAP output objects into memory, and manually attach the additional sample metadata to each object.

```r
#  Load required R packages
library("bsseq")
library("HDF5Array")
library("ggplot2")

#  Path to the sample metadata and BiocMAP outputs. The outputs are too large
#  to host in this repository, so we reference local paths here
meta_file <- file.path(
    "/dcl02/lieber/ajaffe/FlowRNA_RNAseq/WGBS",
    "FlowRNA_WGBS_Sample_Information_with_Pheno_Info.csv"
)
out_dir <- file.path(
    "/dcs04/lieber/lcolladotor/flowRNA_LIBD001/flowRNA_WGBS/processed-data",
    "03_BiocMAP/BiocMAP_output"
)
```

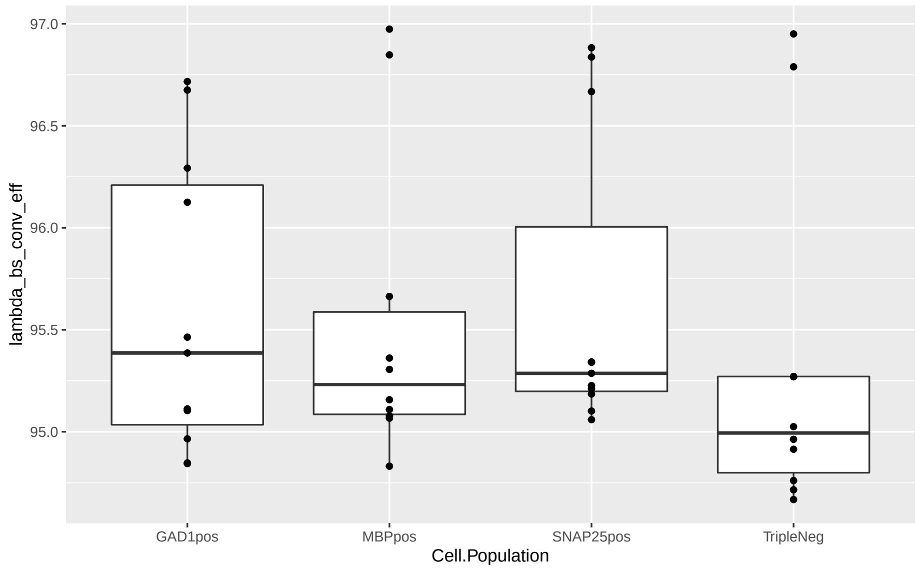
```r
#  Load the 'CpG'-context object
bs_cpg <- loadHDF5SummarizedExperiment(
    file.path(out_dir, "BSobjects", "objects", "combined"),
    prefix = "CpG"
)

#  Load the 'CpH'-context object. Note: this requires quite a bit of memory
#  (~23GB) even though the assays are disk-backed!
bs_cph <- loadHDF5SummarizedExperiment(
    file.path(out_dir, "BSobjects", "objects", "combined"),
    prefix = "CpH"
)

#  Read in experiment-specific metadata and ensure sample ID orders match
meta <- read.csv(meta_file)
meta <- meta[match(colnames(bs_cpg), meta$LIBD.), ]

#  Add this metadata to the Bioconductor objects
colData(bs_cpg) <- cbind(colData(bs_cpg), meta)
colData(bs_cph) <- cbind(colData(bs_cph), meta)

#  Keep a copy of the metadata as a data frame, for easy plotting
meta_df <- data.frame(colData(bs_cpg))
```

2 Exploratory Plots

2.1 Bisulfite-Conversion Efficiency by Cell Population

This experiment used spike-ins of the lambda bacteriophage genome, which were quantified via BiocMAP to infer bisulfite-conversion rate. Successful bisulfite conversion is a pre-requisite for accurate methylation calls, so we'd like to see both that values are close to 1, and that values are not significantly different by sample (or by sample-related variables like cell population). We'll explore this visually below.

```
ggplot(meta_df, aes(x = Cell.Population, y = lambda_bs_conv_eff)) +
    geom_boxplot(outlier.shape = NA) +
    geom_point()
## Warning: Removed 1 rows containing non-finite values (stat_boxplot).
## Warning: Removed 1 rows containing missing values (geom_point).
```

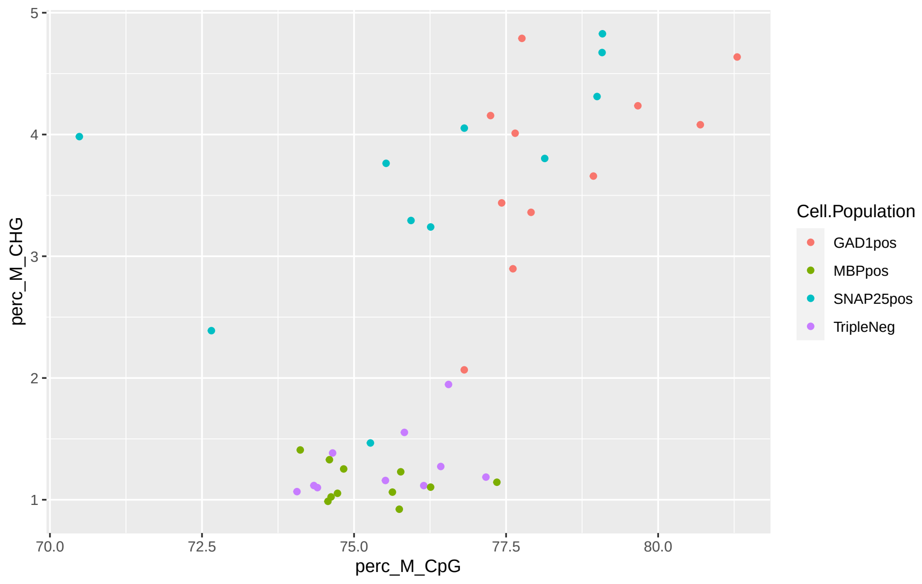


2.2 Relationship between Methylation Fractions across Cytosine Context, by Cell Population

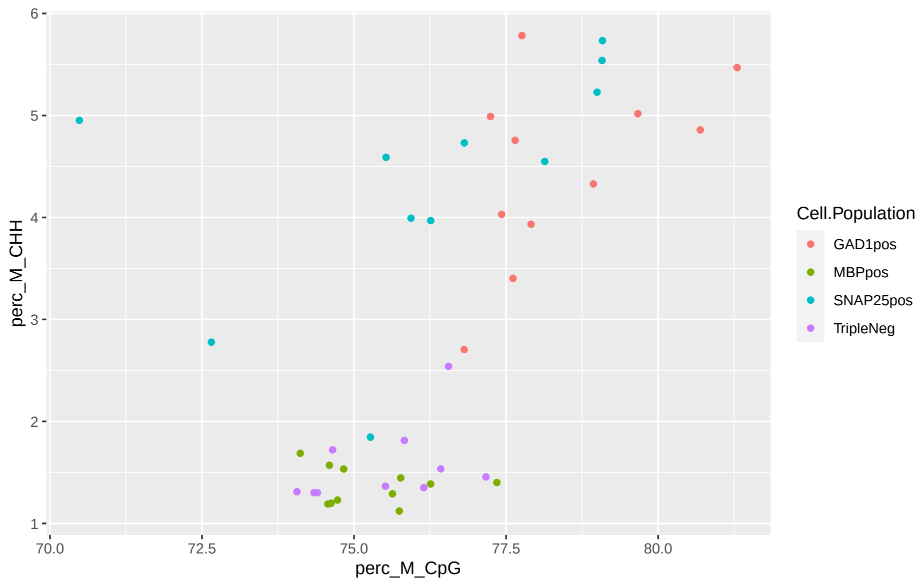
Next, we'll explore if average methylation rate for each cytosine context correlates with that of other contexts across sample. For example, is a sample with highly methylated CpGs likely to have highly methylated CHGs (the first plot)?

We observe a few interesting facts; first, there is a visibly obvious correlation between average methylation rates of different cytosine contexts by sample. This is highly pronounced between CpH contexts (CHG vs. CHH). In each case, the relation appears roughly linear, though this is questionable for CpG vs. CHH context comparison. Another observation is that samples tend to cluster fairly well by cell population. Finally, for comparisons of CpG vs. CpH context, the strength of correlation between methylation rates varies significantly by cell type, with `MBPpos` and `TripleNeg` showing only weak correlation at best.

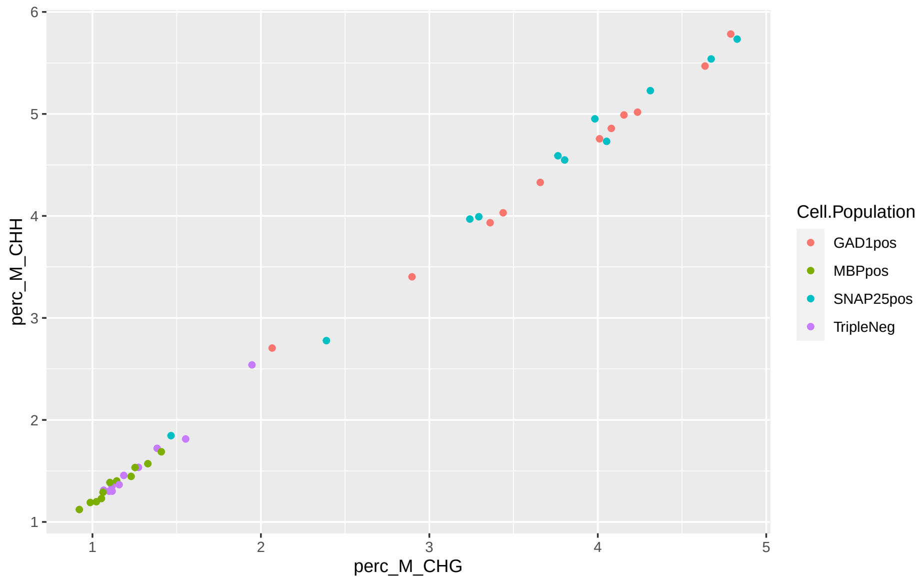
```
#  Clean this up: code should be written around the idea we are plotting each
#  combination of methylation fractions (CpG, CHG, CHH), and plots should be
#  placed in a single grid

ggplot(meta_df, aes(x = perc_M_CpG, y = perc_M_CHG, color = Cell.Population)) +
    geom_point()
```



```
ggplot(meta_df, aes(x = perc_M_CpG, y = perc_M_CHH, color = Cell.Population)) +
    geom_point()
```



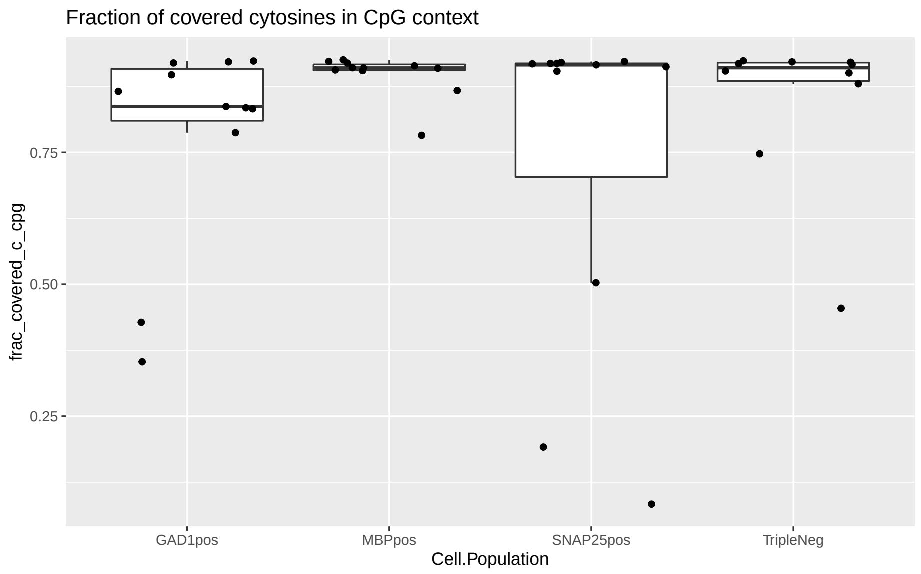
```
ggplot(meta_df, aes(x = perc_M_CHG, y = perc_M_CHH, color = Cell.Population)) +
    geom_point()
```



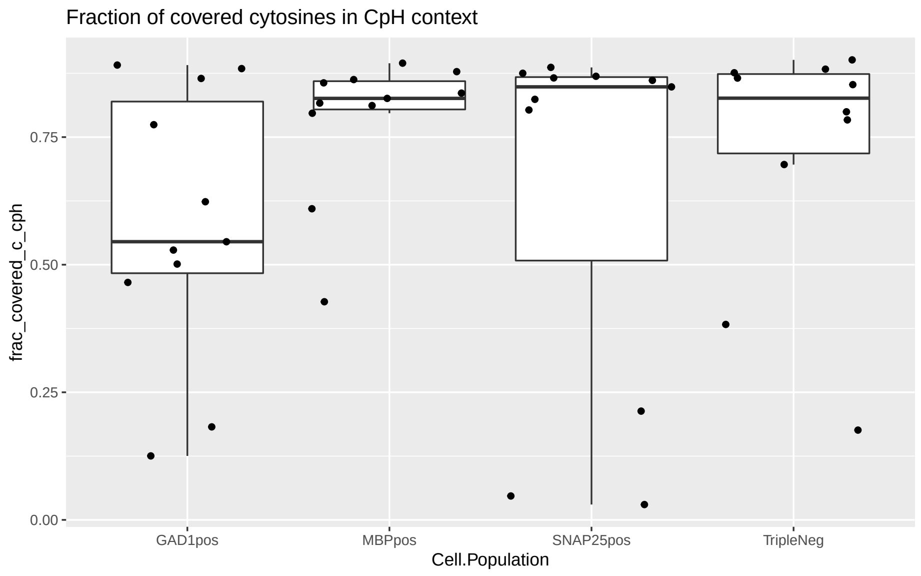
2.3 Fraction of Covered Cytosines by Cell Population

Another useful piece of information is how well-covered the genome is with methylation information. Does coverage of cytosines vary by a sample's cell type?

```
#  The matrices in 'assays(bs_cpg)' and 'assays(bs_cph)' are stored on disk. To
#  speed up some below computations, we raise the per-block memory size
setAutoBlockSize(1e9)
## automatic block size set to 1e+09 bytes (was 1e+08)

#  Get the proportion of cytosines in each object (context) that have non-zero
#  coverage in at least one sample
meta_df$frac_covered_c_cpg <- DelayedArray::colMeans(assays(bs_cpg)$Cov > 3)
meta_df$frac_covered_c_cph <- DelayedArray::colMeans(assays(bs_cph)$Cov > 3)

#   Plot fraction of covered CpG-context cytosines by cell population
ggplot(meta_df, aes(x = Cell.Population, y = frac_covered_c_cpg)) +
    geom_boxplot(outlier.shape = NA) +
    geom_point(position = "jitter") +
    labs(title = "Fraction of covered cytosines in CpG context")
```



```
#   Plot fraction of covered CpH-context cytosines by cell population
ggplot(meta_df, aes(x = Cell.Population, y = frac_covered_c_cph)) +
    geom_boxplot(outlier.shape = NA) +
    geom_point(position = "jitter") +
    labs(title = "Fraction of covered cytosines in CpH context")
```

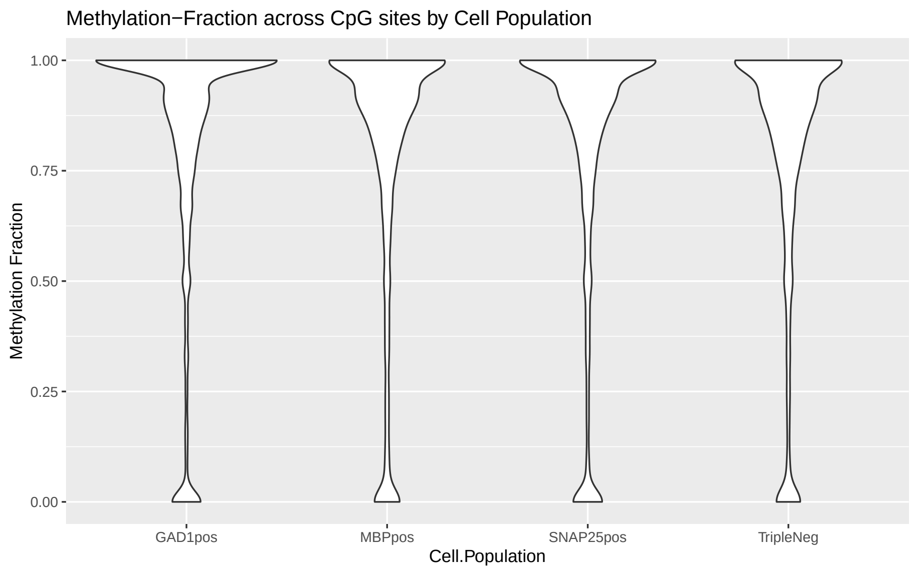


2.4 Distribution of Methylation Fractions across Cytosines by Cell Population

Grouping together all samples of a particular cell type, we'll explore the methylation-fraction distribution across cytosines, separately for both CpG and CpH contexts.

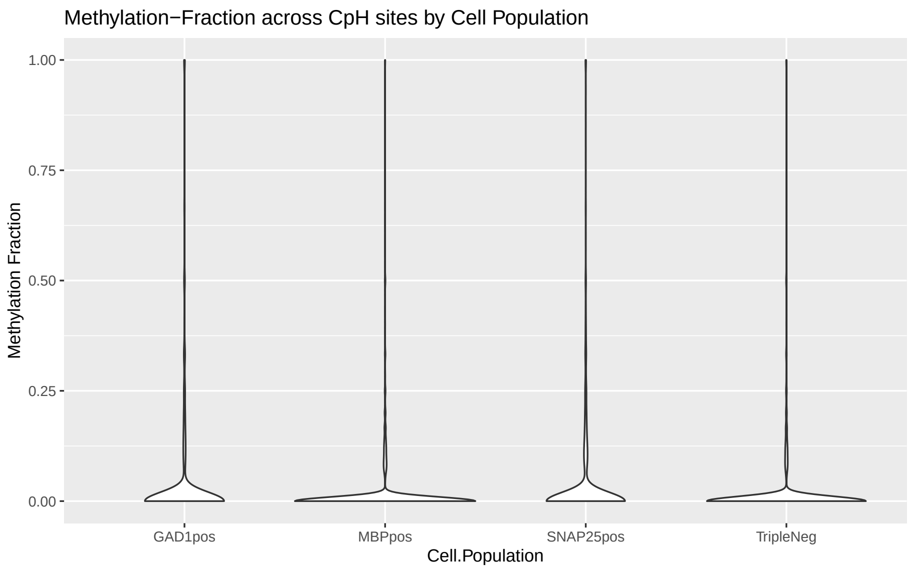
For both cytosine contexts, we observe a bimodal distribution with peaks at fractions of 0 and 1. This suggests that within a particular sample, a cytosine site is disproportionately likely to have consistent methylation pattern. For example, many sites are such that all observations of the cytosine are methylated for a particular sample. Similarly, we don't see many sites where around half of the observed site are methylated for a given sample. It's also worth noting that the apparent bimodal form is likely not an artifact of low coverage– i.e., only a small fraction of sites are covered just once or twice, a circumstance that would cause over-representation of the fractions of 0 or 1.

```
#   Randomly subset to a particular number of cytosines, to both control memory
#   and speed up plotting
max_sites <- 1000

#   Look at CpG sites first
indices <- sample(nrow(bs_cpg), max_sites)
m_frac <- assays(bs_cpg)$M[indices, ] / assays(bs_cpg)$Cov[indices, ]

#   It's worth looking at the distribution of coverage by site, since in theory
#   this could be cause for the bimodality observed in the plot below
table(assays(bs_cpg)$Cov[indices, ])
##
##    0    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15
## 3456 1558 1455 1500 1427 1546 1621 1689 1756 1872 1952 1935 2058 2068 1969 2004
##   16   17   18   19   20   21   22   23   24   25   26   27   28   29   30   31
## 1949 1752 1571 1392 1220 1039  900  708  563  508  375  244  208  179  128   82
##   32   33   34   35   36   37   38   39   40   41   42   43   44   45   46   47
##   80   58   34   28   13    6    6    5    8    2    1    5    2    2    1    1
##   48   52   53   56   58   59   61   63   66   71   72   73   74   76   77   79
##    1    1    3    2    1    2    2    1    1    1    1    2    1    1    1    1
##   82   85   86   87   88   95   96   99  101  102  104  105  110  111  113  114
##    1    2    2    1    2    1    1    1    1    1    1    1    2    2    1    1
##  115  124  128  132  147  158  164  170  173  175  177  185  186  189  192  195
##    1    1    1    1    2    1    1    1    1    1    1    1    1    1    1    1
##  235  236  237  285
##    1    1    1    1

#   Form a data frame for easy plotting: we'll collapse methylation data for all
#   samples into a single column, 'm_frac'. Here 'LIBD.' denotes sample ID
meth_df <- data.frame(
    "m_frac" = as.numeric(m_frac),
    "LIBD." = rep(colnames(m_frac), each = max_sites)
)

#   Label each observation (methylation fraction for a particular cytosine) with
#   the cell population of the associated sample
meth_df$Cell.Population <- meta_df$Cell.Population[
    match(meth_df$"LIBD.", meta_df$"LIBD.")
]

ggplot(meth_df, aes(x = Cell.Population, y = m_frac)) +
    geom_violin() +
    labs(
        title = "Methylation-Fraction across CpG sites by Cell Population",
        y = "Methylation Fraction"
    )
## Warning: Removed 3456 rows containing non-finite values (stat_ydensity).
```



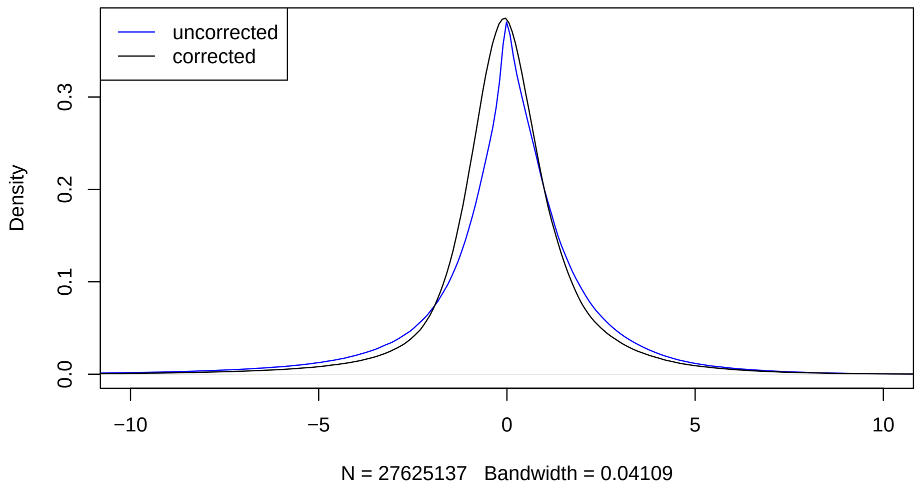
```
#   Now look at CpH sites
indices <- sample(nrow(bs_cph), max_sites)
m_frac <- assays(bs_cph)$M[indices, ] / assays(bs_cph)$Cov[indices, ]

#   Again, we'll look at the distribution of coverage by site, since in theory
#   this could be cause for the bimodality observed in the plots below
table(assays(bs_cph)$Cov[indices, ])
##
##    0    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15
## 4969 2762 2856 3178 3342 3547 3665 3576 3391 2905 2379 1869 1532 1021  692  498
##   16   17   18   19   20   21   22   23   24   25   26   27   28   29   30   31
##  315  186  115   77   45   31   16    8    6    3    1    5    3    1    1    1
##   34   36   42
##    2    1    1

#   Form a data frame for easy plotting: we'll collapse methylation data for all
#   samples into a single column, 'm_frac'. Here 'LIBD.' denotes sample ID
meth_df <- data.frame(
    "m_frac" = as.numeric(m_frac),
    "LIBD." = rep(colnames(m_frac), each = max_sites)
)

#   Label each observation (methylation fraction for a particular cytosine) with
#   the cell population of the associated sample
meth_df$Cell.Population <- meta_df$Cell.Population[
    match(meth_df$"LIBD.", meta_df$"LIBD.")
]
```

```
ggplot(meth_df, aes(x = Cell.Population, y = m_frac)) +
    geom_violin() +
    labs(
        title = "Methylation-Fraction across CpH sites by Cell Population",
        y = "Methylation Fraction"
    )
## Warning: Removed 4969 rows containing non-finite values (stat_ydensity).
```



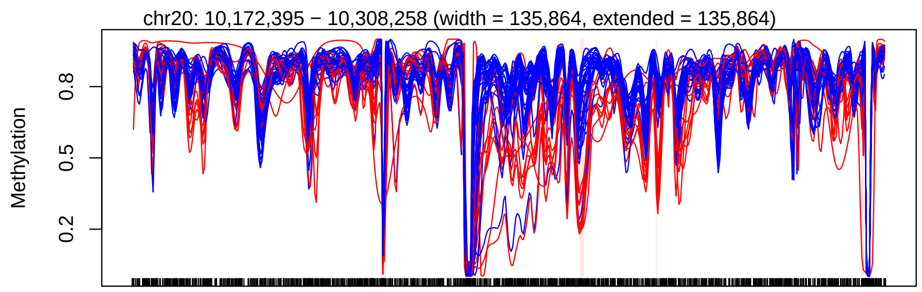
```
BiocParallel::register(BiocParallel::MulticoreParam(1))

cell_pop <- "SNAP25pos"

#   To avoid false positive DMRs, we'll subset to CpGs where at least 5 samples
#   in each group have at least two observations of the given CpG
num_cov_samples <- 5
num_cov_count <- 2

#   Subset object based on coverage requirements
bs_cov <- getCoverage(bs_cpg)
loci_to_keep <- which(
    rowSums(
        bs_cov[, bs_cpg$Cell.Population == cell_pop] >= num_cov_count
    ) >= num_cov_samples &
        rowSums(
            bs_cov[, bs_cpg$Cell.Population != cell_pop] >= num_cov_count
        ) >= num_cov_samples
)
length(loci_to_keep)
## [1] 27625137

bs_cpg_sub <- bs_cpg[loci_to_keep, ]
```

```
#   Define the two cell-population-based groups used to compute t-statistics
group1 <- unique(bs_cpg_sub$"LIBD."[bs_cpg_sub$Cell.Population == cell_pop])
group2 <- unique(bs_cpg_sub$"LIBD."[bs_cpg_sub$Cell.Population != cell_pop])

#   Compute the t-stat and show the marginal distribution
cpg_t_stat <- BSmooth.tstat(
    bs_cpg_sub,
    group1 = group1, group2 = group2,
    estimate.var = "group2", local.correct = TRUE, verbose = TRUE
)
## [BSmooth.tstat] preprocessing ... done in 35.1 sec
## [BSmooth.tstat] computing stats within groups ... done in 367.8 sec
## [BSmooth.tstat] computing stats across groups ... done in 532.7 sec
plot(cpg_t_stat)
```



```
#   Grab the genomic range associated with this particular cell population
if (cell_pop == "SNAP25pos") {
    gene_range <- GRanges("chr20:10172395-10308258")
} else if (cell_pop == "MBPpos") {
    gene_range <- GRanges("chr18:76978827-77133708")
} else if (cell_pop == "GAD1pos") {
    gene_range <- GRanges("chr2:170813210-170861151")
} else {
    stop(
        paste0("No gene associated with this cell population '", cell_pop, "'.")
    )
}

t_stat_df <- getStats(cpg_t_stat)

#   Filters for the upcoming DMRs: minimum magnitude of t-stat and min number of
#   base pairs, respectively
thres <- sd(t_stat_df[, "tstat.corrected"]) * 2.5
n_bases <- 3
```

```
abs_mean_diff <- 0.1

#   Compute DMRs and apply above filters
dmrs_orig <- dmrFinder(cpg_t_stat, cutoff = c(-1 * thres, thres))
## [dmrFinder] creating dmr data.frame
dmrs <- subset(dmrs_orig, n >= n_bases & abs(meanDiff) >= abs_mean_diff)

#   Color plots by cell population group
p_data <- pData(bs_cpg_sub)
p_data$col <- ifelse(bs_cpg_sub$Cell.Population == cell_pop, "red", "blue")
pData(bs_cpg_sub) <- p_data

#   Plot the region around the gene associated with this particular cell
#   population
plotRegion(
    bs_cpg_sub, gene_range,
    extend = width(gene_range), addRegions = dmrs
)
```



Date the vignette was generated:

```
## [1] "2022-02-18 13:56:56 EST"
```

Wallclock time spent generating the vignette:

```
## Time difference of 3.265 hours
```

R session information:

```
## - Session info -----------------------------------------------------------------------------------------------
##  setting  value
##  version  R version 4.1.2 Patched (2021-11-04 r81138)
##  os       CentOS Linux 7 (Core)
##  system   x86_64, linux-gnu
##  ui       X11
##  language (EN)
##  collate  en_US.UTF-8
##  ctype    en_US.UTF-8
##  tz       US/Eastern
##  date     2022-02-18
##  pandoc   2.13 @ /jhpce/shared/jhpce/core/conda/miniconda3-4.6.14/envs/svnR-4.1.x/bin/ (via rmarkdown)
##
## - Packages ---------------------------------------------------------------------------------------------------
##  package              * version  date (UTC) lib source
##  assertthat             0.2.1    2019-03-21 [2] CRAN (R 4.1.0)
##  Biobase              * 2.54.0   2021-10-26 [2] Bioconductor
##  BiocGenerics         * 0.40.0   2021-10-26 [2] Bioconductor
##  BiocIO                 1.4.0    2021-10-26 [2] Bioconductor
##  BiocManager            1.30.16  2021-06-15 [2] CRAN (R 4.1.2)
##  BiocParallel           1.28.3   2021-12-09 [2] Bioconductor
##  BiocStyle            * 2.22.0   2021-10-26 [1] Bioconductor
##  Biostrings             2.62.0   2021-10-26 [2] Bioconductor
##  bitops                 1.0-7    2021-04-24 [2] CRAN (R 4.1.0)
##  bookdown               0.24     2021-09-02 [1] CRAN (R 4.1.2)
##  BSgenome               1.62.0   2021-10-26 [2] Bioconductor
##  bsseq                * 1.30.0   2021-10-26 [2] Bioconductor
##  cli                    3.2.0    2022-02-14 [2] CRAN (R 4.1.2)
##  colorspace             2.0-2    2021-06-24 [2] CRAN (R 4.1.0)
##  crayon                 1.5.0    2022-02-14 [2] CRAN (R 4.1.2)
##  data.table             1.14.2   2021-09-27 [2] CRAN (R 4.1.2)
##  DBI                    1.1.2    2021-12-20 [2] CRAN (R 4.1.2)
##  DelayedArray         * 0.20.0   2021-10-26 [2] Bioconductor
##  DelayedMatrixStats     1.16.0   2021-10-26 [2] Bioconductor
##  digest                 0.6.29   2021-12-01 [2] CRAN (R 4.1.2)
##  dplyr                  1.0.8    2022-02-08 [2] CRAN (R 4.1.2)
##  ellipsis               0.3.2    2021-04-29 [2] CRAN (R 4.1.0)
##  evaluate               0.14     2019-05-28 [2] CRAN (R 4.1.0)
##  fansi                  1.0.2    2022-01-14 [2] CRAN (R 4.1.2)
##  farver                 2.1.0    2021-02-28 [2] CRAN (R 4.1.0)
##  fastmap                1.1.0    2021-01-25 [2] CRAN (R 4.1.0)
##  generics               0.1.2    2022-01-31 [2] CRAN (R 4.1.2)
##  GenomeInfoDb         * 1.30.1   2022-01-30 [2] Bioconductor
##  GenomeInfoDbData       1.2.7    2021-11-01 [2] Bioconductor
##  GenomicAlignments      1.30.0   2021-10-26 [2] Bioconductor
##  GenomicRanges        * 1.46.1   2021-11-18 [2] Bioconductor
##  ggplot2              * 3.3.5    2021-06-25 [2] CRAN (R 4.1.0)
##  glue                   1.6.1    2022-01-22 [2] CRAN (R 4.1.2)
##  gtable                 0.3.0    2019-03-25 [2] CRAN (R 4.1.0)
##  gtools                 3.9.2    2021-06-06 [2] CRAN (R 4.1.0)
##  HDF5Array            * 1.22.1   2021-11-14 [2] Bioconductor
##  htmltools              0.5.2    2021-08-25 [2] CRAN (R 4.1.2)
##  httr                   1.4.2    2020-07-20 [2] CRAN (R 4.1.0)
##  IRanges              * 2.28.0   2021-10-26 [2] Bioconductor
##  jsonlite               1.7.3    2022-01-17 [2] CRAN (R 4.1.2)
##  knitr                  1.37     2021-12-16 [2] CRAN (R 4.1.2)
##  labeling               0.4.2    2020-10-20 [2] CRAN (R 4.1.0)
##  lattice                0.20-45  2021-09-22 [3] CRAN (R 4.1.2)
##  lifecycle              1.0.1    2021-09-24 [2] CRAN (R 4.1.2)
##  limma                  3.50.0   2021-10-26 [2] Bioconductor
##  locfit                 1.5-9.4  2020-03-25 [2] CRAN (R 4.1.0)
##  lubridate              1.8.0    2021-10-07 [2] CRAN (R 4.1.2)
##  magrittr               2.0.2    2022-01-26 [2] CRAN (R 4.1.2)
##  Matrix               * 1.4-0    2021-12-08 [3] CRAN (R 4.1.2)
##  MatrixGenerics       * 1.6.0    2021-10-26 [2] Bioconductor
##  matrixStats          * 0.61.0   2021-09-17 [2] CRAN (R 4.1.2)
##  munsell                0.5.0    2018-06-12 [2] CRAN (R 4.1.0)
##  permute                0.9-7    2022-01-27 [2] CRAN (R 4.1.2)
##  pillar                 1.7.0    2022-02-01 [2] CRAN (R 4.1.2)
##  pkgconfig              2.0.3    2019-09-22 [2] CRAN (R 4.1.0)
##  plyr                   1.8.6    2020-03-03 [2] CRAN (R 4.1.0)
##  purrr                  0.3.4    2020-04-17 [2] CRAN (R 4.1.0)
##  R.methodsS3            1.8.1    2020-08-26 [2] CRAN (R 4.1.0)
##  R.oo                   1.24.0   2020-08-26 [2] CRAN (R 4.1.0)
##  R.utils                2.11.0   2021-09-26 [2] CRAN (R 4.1.2)
##  R6                     2.5.1    2021-08-19 [2] CRAN (R 4.1.2)
##  Rcpp                   1.0.8    2022-01-13 [2] CRAN (R 4.1.2)
##  RCurl                  1.98-1.6 2022-02-08 [2] CRAN (R 4.1.2)
##  RefManageR           * 1.3.0    2020-11-13 [1] CRAN (R 4.1.2)
##  restfulr               0.0.13   2017-08-06 [2] CRAN (R 4.1.0)
##  rhdf5                * 2.38.0   2021-10-26 [2] Bioconductor
##  rhdf5filters           1.6.0    2021-10-26 [2] Bioconductor
##  Rhdf5lib               1.16.0   2021-10-26 [2] Bioconductor
##  rjson                  0.2.21   2022-01-09 [2] CRAN (R 4.1.2)
##  rlang                  1.0.1    2022-02-03 [2] CRAN (R 4.1.2)
##  rmarkdown              2.11     2021-09-14 [2] CRAN (R 4.1.2)
##  Rsamtools              2.10.0   2021-10-26 [2] Bioconductor
##  rtracklayer            1.54.0   2021-10-26 [2] Bioconductor
##  S4Vectors            * 0.32.3   2021-11-21 [2] Bioconductor
##  scales                 1.1.1    2020-05-11 [2] CRAN (R 4.1.0)
##  sessioninfo          * 1.2.2    2021-12-06 [2] CRAN (R 4.1.2)
##  sparseMatrixStats      1.6.0    2021-10-26 [2] Bioconductor
##  stringi                1.7.6    2021-11-29 [2] CRAN (R 4.1.2)
##  stringr                1.4.0    2019-02-10 [2] CRAN (R 4.1.0)
##  SummarizedExperiment * 1.24.0   2021-10-26 [2] Bioconductor
##  tibble                 3.1.6    2021-11-07 [2] CRAN (R 4.1.2)
##  tidyselect             1.1.1    2021-04-30 [2] CRAN (R 4.1.0)
##  utf8                   1.2.2    2021-07-24 [2] CRAN (R 4.1.0)
##  vctrs                  0.3.8    2021-04-29 [2] CRAN (R 4.1.0)
##  withr                  2.4.3    2021-11-30 [2] CRAN (R 4.1.2)
##  xfun                   0.29     2021-12-14 [2] CRAN (R 4.1.2)
##  XML                    3.99-0.8 2021-09-17 [2] CRAN (R 4.1.2)
##  xml2                   1.3.3    2021-11-30 [2] CRAN (R 4.1.2)
##  XVector                0.34.0   2021-10-26 [2] Bioconductor
##  yaml                   2.3.4    2022-02-17 [2] CRAN (R 4.1.2)
##  zlibbioc               1.40.0   2021-10-26 [2] Bioconductor
##
##  [1] /users/neagles/R/4.1.x
##  [2] /jhpce/shared/jhpce/core/conda/miniconda3-4.6.14/envs/svnR-4.1.x/R/4.1.x/lib64/R/site-library
##  [3] /jhpce/shared/jhpce/core/conda/miniconda3-4.6.14/envs/svnR-4.1.x/R/4.1.x/lib64/R/library
##
## ------------------------------------------------------------------------------------------------
```

3 Bibliography

This vignette was generated using *BiocStyle* (Oleś, 2021) with *knitr* (Xie, 2021) and *rmarkdown* (Allaire, Xie, McPherson, Luraschi, Ushey, Atkins, Wickham, Cheng, Chang, and Iannone, 2021) running behind the scenes.

Citations made with *RefManageR* (McLean, 2017).

[1] J. Allaire, Y. Xie, J. McPherson, et al. rmarkdown: Dynamic Documents for R. R package version 2.11. 2021. URL: https://github.com/rstudio/rmarkdown.

[2] M. W. McLean. "RefManageR: Import and Manage BibTeX and BibLaTeX References in R". In: The Journal of Open Source Software (2017). DOI: 10.21105/joss.00338.

[3] A. Oleś. BiocStyle: Standard styles for vignettes and other Bioconductor documents. R package version 2.22.0. 2021. URL: https://github.com/Bioconductor/BiocStyle.

[4] Y. Xie. knitr: A General-Purpose Package for Dynamic Report Generation in R. R package version 1.37. 2021. URL: https://yihui.org/knitr/.