DNA methylation tutorial: Bisulfite-seq Data Analysis with bsseq and dmrseq

by
Roseric Azondekon, PhD
University of Wisconsin Milwaukee

June 12, 2019

Background

In a previous tutorial, we showed you how to download and process Bisulfite-seq DNA methylation FASTQ files for read alignment on a reference sequence. In this tutorial, we show you how to run DNA methylation analysis using the `bsseq` and `dmrseq` package in `R`.

We set our working directory to the `tuto` folder created in our first tutorial.

```
In [ ]: setwd('./tuto')
```

Now, let's install all the required packages for this tutorial.

```
In [ ]: # Indicate package repositories to R...
        repositories <- c("https://cloud.r-project.org",
                          "https://bioconductor.org/packages/3.7/bioc",
                          "https://bioconductor.org/packages/3.7/data/annotation",
                          "https://bioconductor.org/packages/3.7/data/experiment",
                          "https://www.stats.ox.ac.uk/pub/RWin",
                          "http://www.omegahat.net/R",
                          "https://R-Forge.R-project.org",
                          "https://www.rforge.net",
                          "https://cloud.r-project.org",
                          "http://www.bioconductor.org",
                          "http://www.stats.ox.ac.uk/pub/RWin")

        # Package list to download
        packages <- c("bsseq","bsseqdata","dmrseq")

        # Install and load missing packages
        new.packages <- packages[!(packages %in% installed.packages()[,"Package"])]

        if(length(new.packages)){
            install.packages(new.packages, repos = repositories)
```

```
}

lapply(packages, require, character.only = TRUE)
```

1 Obtaining methylation data from Bismark extraction methylation calls

We read in the methylation calls directly from the `Bismark` methylation extractor files obtained from the last tutoral. The files are located within the `bismark_methCalls` folder (see previous tutorial). For that purpose, we use the `read.bismark()` function from the `bsseq` package as described below:

```
In [ ]: files_loc <- file.path(getwd(),'bismark_methCalls')
        samples <- list.dirs(files_loc,full.names = F,recursive = F)
        samples
```

```
In [ ]: conditions <- c(rep(c("normal", "cancer"), each = 2))
        sampleData <- data.frame(condition = conditions)
        rownames(sampleData) <- samples
        sampleData
```

```
In [ ]: methyl_files <- list.files(files_loc, "\\cov.gz$",
                                   full.names=TRUE, recursive=TRUE)
        methyl_files
```

```
In [ ]: # Will generate specifically for this set of data, 4 variables:
        # methyl_data1, methyl_data2, methyl_data3, methyl_data4
        for(i in 1:length(methyl_files)){
            sampleTable <- data.frame(condition = conditions[i])
            rownames(sampleTable) <- samples[i]
            assign(
                paste0("methyl_data",i),
                read.bismark(methyl_files[i],
                             loci = NULL,
                             colData = sampleTable,
                             rmZeroCov = FALSE,
                             strandCollapse = TRUE,
                             BPPARAM = bpparam(),
                             BACKEND = "HDF5Array",
                             dir = tempfile("bsseq"),
                             replace = FALSE,
                             chunkdim = NULL,
                             level = NULL,
                             nThread = 8,
                             verbose = getOption("verbose"))
            )
        }
```

```
In [ ]: methyl_data1
        methyl_data2
        methyl_data3
        methyl_data4

In [ ]: pData(methyl_data1)
        pData(methyl_data2)
        pData(methyl_data3)
        pData(methyl_data4)
```

We now combine all methylation data for all 4 samples:

```
In [ ]: combined_data <- combine(methyl_data1, methyl_data2, methyl_data3, methyl_data4)

In [ ]: combined_data

In [ ]: pData(combined_data)
```

2 Smoothing

The first step of the analysis is to smooth the data

```
In [ ]: combined_data.fit <- BSmooth(
            bsseq = combined_data,
            BPPARAM = MulticoreParam(workers = 7),
            verbose = TRUE)
```

Since the previous step is time consuming and computationally expensive, let's save the smoothed data:

```
In [ ]: save(combined_data.fit, file = "combined_data_fit.rda")
```

You may load the `combined_data.fit` by running the following code:

```
In [ ]: load("combined_data_fit.rda")

In [ ]: combined_data.fit

In [ ]: ## The average coverage of CpGs on the two chromosomes
        round(colMeans(getCoverage(combined_data)), 1)

In [ ]: ## Number of CpGs in two chromosomes
        length(combined_data)

In [ ]: ## Number of CpGs which are covered by at least 1 read in all 4 samples
        sum(rowSums(getCoverage(combined_data) >= 1) == 4)

In [ ]: # Number of CpGs with 0 coverage in all samples
        sum(rowSums(getCoverage(combined_data)) == 0)
```

3 Computing t-statistics

To avoid too many differentially methilated regions (DRMs), we remove CpGs with little or no coverage (which are likely false positives). We keep CpGs where at least 1 cancer samples and at least 1 normal samples have at least 2x in coverage.

```
In [ ]: # which loci and sample indices to keep
        keep.index <- which(DelayedMatrixStats::rowSums2(getCoverage(combined_data,
                                                                     type="Cov")==0) == 0)
        sample.index <- which(pData(combined_data)$condition %in% c("normal", "cancer"))

        combined_data.filtered <- combined_data[keep.index, sample.index]
```

```
In [ ]: combined_data.filtered
```

For t-statistics, we will only keep CpGs where at least 2 cancer samples and at least 2 normal samples have at least 2x in coverage.

```
In [ ]: combined_data.cov <- getCoverage(combined_data.fit)
        keep.index2 <- which(rowSums(combined_data.cov[,
                                     combined_data$condition == "cancer"] >= 2) >= 2 &
                             rowSums(combined_data.cov[,
                                     combined_data$condition == "normal"] >= 2) >= 2)
        length(keep.index2)
```

```
In [ ]: combined_data.fit2 <- combined_data.fit[keep.index2]
```

Let's first arrange the two groups for the t-test:

```
In [ ]: # In grp1, we keep all the normal sample names, and
        # in grp2, all the cancer sample names
        grp1 <- rownames(sampleData)[sampleData$condition == 'normal']
        grp2 <- rownames(sampleData)[sampleData$condition == 'cancer']
        grp1
        grp2
```

We now compute t-statistics with the `BSmooth.tstat` function provided by the `bsseq` R package.

```
In [ ]: combined_data.tstat <- BSmooth.tstat(combined_data.fit2,
                                             group1 = grp2,
                                             group2 = grp1,
                                             estimate.var = "group2",
                                             local.correct = TRUE,
                                             mc.cores = 8,
                                             verbose = TRUE)
```

```
In [ ]: combined_data.tstat
```

```
In [ ]: stats <- as.data.frame(combined_data.tstat@stats)
        head(stats)
```

Let's check the marginal distribution of the t-statistic:

```
In [ ]: plot(density(as.numeric(as.vector(stats$tstat.corrected)),
                     na.rm=T), xlim = c(-15, 15), col = "red", main = "")
        lines(density(as.numeric(as.vector(stats$tstat)),
                      na.rm=T), col = "blue")
        legend("topright", legend=c("corrected","uncorrected"),
               col=c("red","blue"), lty=1)
```

The "blocks" of hypomethylation are clearly visible in the marginal distribution of the uncorrected t-statistics.

4 Finding Differentially Methylated Regions (DMRs)

We use the `dmrseq` function of the `dmrseq` R package to compute the DMRs.

```
In [ ]: # run the results for a subset of 60,000 CpGs in the interest of computation time.
        # Run with a single core if it fails on multiple cores (workers=1)
        dmrs <- dmrseq(bs=combined_data.filtered[240001:300000,],
                       cutoff = 0.05,
                       BPPARAM = MulticoreParam(workers = 1),
                       testCovariate="condition")
```

```
In [ ]: show(dmrs)
```

4.1 Explore how many regions were significant

How many regions were significant at the FDR (q-value) cutoff of 0.05?

```
In [ ]: sum(dmrs$qval < 0.05)
```

```
In [ ]: # select just the regions below FDR 0.05 and place in a new data.frame
        sigRegions <- dmrs[dmrs$qval < 0.05,]
```

4.2 Proportion of regions with hyper-methylation

```
In [ ]: sum(sigRegions$stat > 0) / length(sigRegions)
```

To interpret the direction of effect, since `dmrseq` uses alphabetical order of the covariate of interest, the condition `cancer` is the reference category.

4.3 Plot DMRs

```
In [ ]: # get annotations for hg18
        annotation <- getAnnot("hg18")
```

```
In [ ]: plotDMRs(combined_data.filtered,
                  regions=dmrs[1,],
                  testCovariate="condition",
                  annoTrack=annotation)
```

5 Detecting large-scale methylation blocks

In some applications, such as cancer, it is of interest to effectively 'zoom out' in order to detect larger (lower-resolution) methylation blocks on the order of hundreds of thousands to millions of bases.

```
In [ ]: # run the results for a subset of 300,000 CpGs in the interest of computation time.
        # Run with a single core if it fails on multiple cores
        blocks <- dmrseq(bs=combined_data.filtered[120001:420000,],
                         cutoff = 0.05,
                         testCovariate='condition',
                         block = TRUE,
                         BPPARAM = MulticoreParam(workers = 1),
                         minInSpan = 500,
                         bpSpan = 5e4,
                         maxGapSmooth = 1e6,
                         maxGap = 5e3)
```

```
In [ ]: show(blocks)
```

Let's also plot the top methylation block from the block analysis:

```
In [ ]: plotDMRs(combined_data.filtered,
                 regions=blocks[1,],
                 testCovariate="condition",
                 annoTrack=annotation)
```

This last DMR plot concludes this tutorial on DNA methylation analysis with the `bsseq` and `dmrseq` R packages. For more information, feel free to check the official bsseq and dmrseq tutorials.