Methrix tutorial

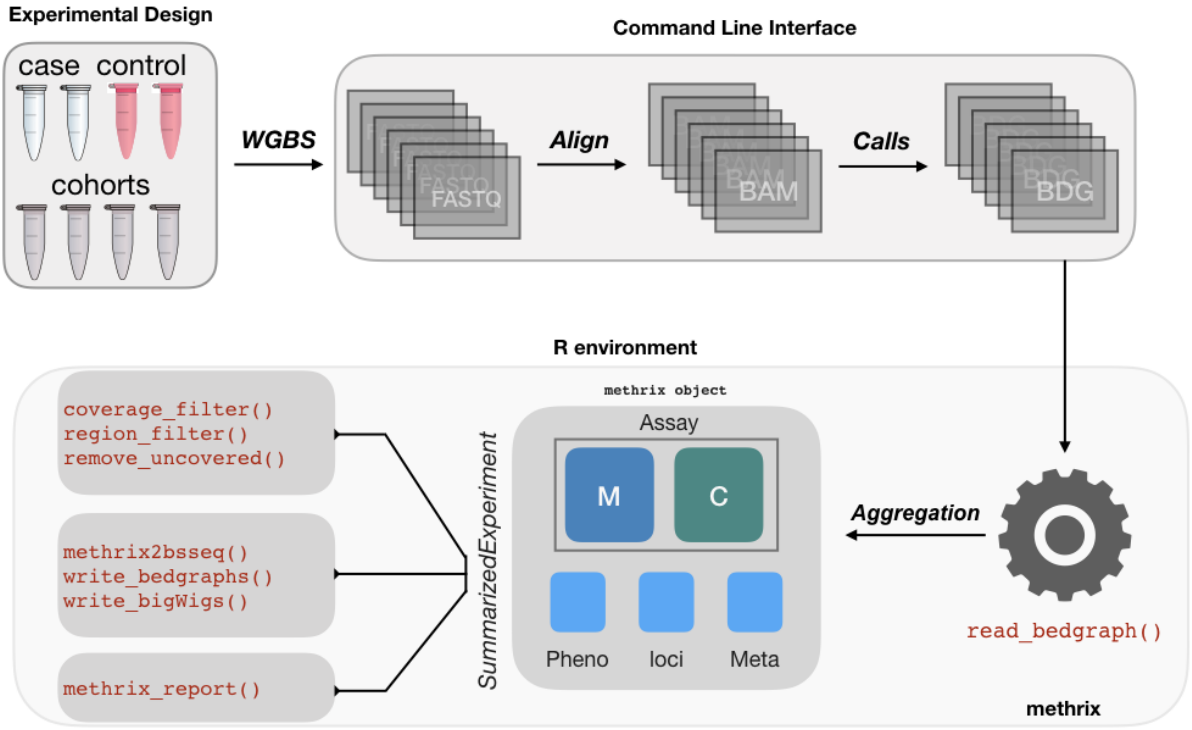
CompEpigen

2019-09-09

Contents

Introduction

"This vignette describes basic usage of the package intended to process several large bedgraph files in R. `Methrix` provides set of function which allows easy importing of various flavours of bedgraphs generated by methylation callers, and many downstream analysis to be performed on large matrices.



Reading bedgraph files

`read_bedgraphs` function is a versatile bedgraph reader intended to import bedgraph files generated virtually by any sort of methylation calling program. It requires user to provide indices for chromosome names, start position and other required fields. There are also presets available to import `bedgraphs` from most common programs such as `Bismark`, `MethylDackel`, and `MethylcTools`.

```
#Load library
library(methrix)
#Genome of your preference to work with
library(BSgenome.Hsapiens.UCSC.hg19)
```

```
#Example bedgraph files
bdg_files = list.files(
  path = system.file('extdata', package = 'methrix'),
  pattern = "*bdg\\.gz$",
  full.names = TRUE
)

print(bdg_files)
#> [1] "C:/Users/tothr/AppData/Local/Temp/Rtmp2vDWG4/temp_libpath1e8052d6d91/methrix/extdata/C1.bdg.gz"
#> [2] "C:/Users/tothr/AppData/Local/Temp/Rtmp2vDWG4/temp_libpath1e8052d6d91/methrix/extdata/C2.bdg.gz"
```

```
#> [3] "C:/Users/tothr/AppData/Local/Temp/Rtmp2vDWG4/temp_libpath1e8052d6d91/methrix/extdata/N1.bdg.gz"
#> [4] "C:/Users/tothr/AppData/Local/Temp/Rtmp2vDWG4/temp_libpath1e8052d6d91/methrix/extdata/N2.bdg.gz"

#Generate some sample annotation table
sample_anno = data.frame(row.names = gsub(pattern = "\\.bdg\\.gz$", replacement = "", x = basename(bdg_

print(sample_anno)
#>    Condition
#> C1    Cancer
#> C2    Cancer
#> N1    Normal
#> N2    Normal
```

We can import bedgraph files with the function `read_bedgraphs` which reads in the bedgraphs, adds CpGs missing from the reference set, and creates a methylation/coverage matrices. If your system has enough memory increasing `vect_batch_size` and `n_threads` speeds up the process. Once the process is complete - it returns an object of class `methrix` which in turn inherits SummarizedExperiment class. `methrix` object contains 'methylation' and 'coverage' matrices (either in-memory or as on-disk HDF5 arrays) along with pheno-data and other basic info. This object can be passed to all downstream functions for various analysis.

Bedgraphs used for vignette are generated with BSMAP from bam files aligned to `mm9` reference genome.

```
#First extract genome wide CpGs from the desired genome
hg19_cpgs = methrix::extract_CPGs(ref_genome = "BSgenome.Hsapiens.UCSC.hg19")
#> -Extracting CpGs
#> -Here is a Chuck Norris joke while you wait..
#> ---------------------------------------------
#> Chuck Norris' penis is a third degree blackbelt, and an honorable 32nd-degree mason.
#> ---------------------------------------------
#> -Done. Extracted 28,700,086 CpGs from 93 contigs.
```

```
#Read the files
meth = methrix::read_bedgraphs(
  files = bdg_files,
  ref_cpgs = hg19_cpgs,
  chr_idx = 1,
  start_idx = 2,
  M_idx = 3,
  U_idx = 4,
  stranded = TRUE,
  collapse_strands = TRUE,
  coldata = sample_anno
)
#> ----------------------------
#> -Preset:        Custom
#> --Missing beta and coverage info. Estimating them from M and U values
#> -CpGs raw:      28,700,086
#> -CpGs filtered: 28,217,448
#> -CpGs stranded: 56,434,896
#> ----------------------------
#> -Batch:         1/1
#> -Processing:    C1.bdg.gz
#> Registered S3 method overwritten by 'R.oo':
#>   method        from
```

```
#>   throw.default R.methodsS3
#> --CpGs missing: 56,434,254
#> -Processing:    C2.bdg.gz
#> --CpGs missing: 56,434,231
#> -Processing:    N1.bdg.gz
#> --CpGs missing: 56,434,233
#> -Processing:    N2.bdg.gz
#> --CpGs missing: 56,434,233
#> -Finished in:   00:03:05 elapsed (00:03:36 cpu)
```

Note: Use argument `pipeline` if your bedgraphs are generated with "Bismark", "MethylDeckal", or "MethylcTools".

```
#Typing meth shows basic summary.
meth
#> An object of class methrix
#>    n_CpGs: 28,217,448
#> n_samples: 4
#>     is_h5: FALSE
#> Reference: hg19
```

HTML QC report

Get basic summary statistics of the `methrix` object with `methrix_report` function which produces an interactive html report

```
methrix::methrix_report(meth = meth, output_dir = tempdir())
```

Filtering

Remove uncovered loci

Usual task in analysis involves removing uncovered CpGs. i.e, those loci which are not covered across all sample (in other words covered only in subset of samples resulting `NA` for rest of the samples).

```
meth = methrix::remove_uncovered(m = meth)
#> -Removed 28,216,749 [100%] uncovered loci of 28,217,448 sites
#> -Finished in:   17.2s elapsed (20.8s cpu)
meth
#> An object of class methrix
#>    n_CpGs: 699
#> n_samples: 4
#>     is_h5: FALSE
#> Reference: hg19
```

Basic operations

Extract methylation/coverage matrices

```
#Coverage matrix
coverage_mat = methrix::get_matrix(m = meth, type = "C")
head(coverage_mat)
#>      C1 C2 N1 N2
#> [1,]  5  6  4  5
#> [2,]  9 11  5  7
#> [3,] 12 12  4 11
#> [4,]  9  7  5 11
#> [5,]  1  1  2  1
#> [6,]  5 12  7 11
```

```
#Methylation matrix
meth_mat = methrix::get_matrix(m = meth, type = "M")
head(meth_mat)
#>             C1         C2        N1        N2
#> [1,] 0.4000000 0.00000000 1.0000000 1.0000000
#> [2,] 0.5555556 0.27272727 0.8000000 0.7142857
#> [3,] 0.3333333 0.25000000 1.0000000 0.9090909
#> [4,] 0.3333333 0.42857143 1.0000000 1.0000000
#> [5,] 0.0000000 1.00000000 0.5000000 0.0000000
#> [6,] 0.0000000 0.08333333 0.7142857 0.4545455
```

```
#If you prefer you can attach loci info to the matrix
meth_mat_with_loci = methrix::get_matrix(m = meth, type = "M", add_loci = TRUE)
meth_mat_with_loci
#>        chr    start strand        C1        C2        N1        N2
#>   1: chr21 27867971      * 0.4000000 0.0000000 1.0000000 1.0000000
#>   2: chr21 27867985      * 0.5555556 0.2727273 0.8000000 0.7142857
#>   3: chr21 27868007      * 0.3333333 0.2500000 1.0000000 0.9090909
#>   4: chr21 27868022      * 0.3333333 0.4285714 1.0000000 1.0000000
#>   5: chr21 27868103      * 0.0000000 1.0000000 0.5000000 0.0000000
#>  ---
#> 695: chr22 49010268      * 0.4000000 0.4000000 0.6666667 0.7500000
#> 696: chr22 49010299      * 0.5000000 0.0000000        NA 1.0000000
#> 697: chr22 49010323      * 0.1428571 0.3000000 0.7500000 0.9090909
#> 698: chr22 49010343      * 0.2500000 0.1818182 0.6250000 1.0000000
#> 699: chr22 49010383      * 0.8000000 0.0000000 0.5000000 0.4000000
```

Coverage filter

Furthermore if you prefer you can filter sites based on coverage conditions.

```
#e.g; Retain all loci which are covered at-least in two sample by 3 or more reads
methrix::coverage_filter(m = meth, cov_thr = 3, min_samples = 2)
#> -Retained 605 of 699 sites
#> -Finished in:   1.080s elapsed (0.700s cpu)
#> An object of class methrix
#>    n_CpGs: 605
#> n_samples: 4
#>     is_h5: FALSE
#> Reference: hg19
```

Subset operations

Subset operations in `methrix` make use of `data.table`s fast binary search which is several orders faster than `bsseq` or other similar packages.

Subset by chromosome

```
#Retain sites only from chromosme chr21
methrix::subset_methrix(m = meth, contigs = "chr21")
#> -Subsetting by contigs
#> An object of class methrix
#>    n_CpGs: 420
#> n_samples: 4
#>     is_h5: FALSE
#> Reference: hg19
```

Subset by genomic regions

Regions can be data.table or GRanges format.

```
#e.g; Retain sites only in TP53 loci
target_loci = data.table::data.table(chr = "chr21", start = 27867971, end =  27868103)
print(target_loci)
#>      chr    start      end
#> 1: chr21 27867971 27868103

methrix::subset_methrix(m = meth, regions = target_loci)
#> -Subsetting by genomic regions
#> An object of class methrix
#>    n_CpGs: 4
#> n_samples: 4
#>     is_h5: FALSE
#> Reference: hg19
```

Subset by samples

```
methrix::subset_methrix(m = meth, samples = "C1")
#> Subsetting by samples
#> An object of class methrix
#>    n_CpGs: 699
#> n_samples: 1
#>     is_h5: FALSE
#> Reference: hg19

#Or you could use [] operator to subset by index
meth[,1]
#> An object of class methrix
#>    n_CpGs: 699
#> n_samples: 1
```

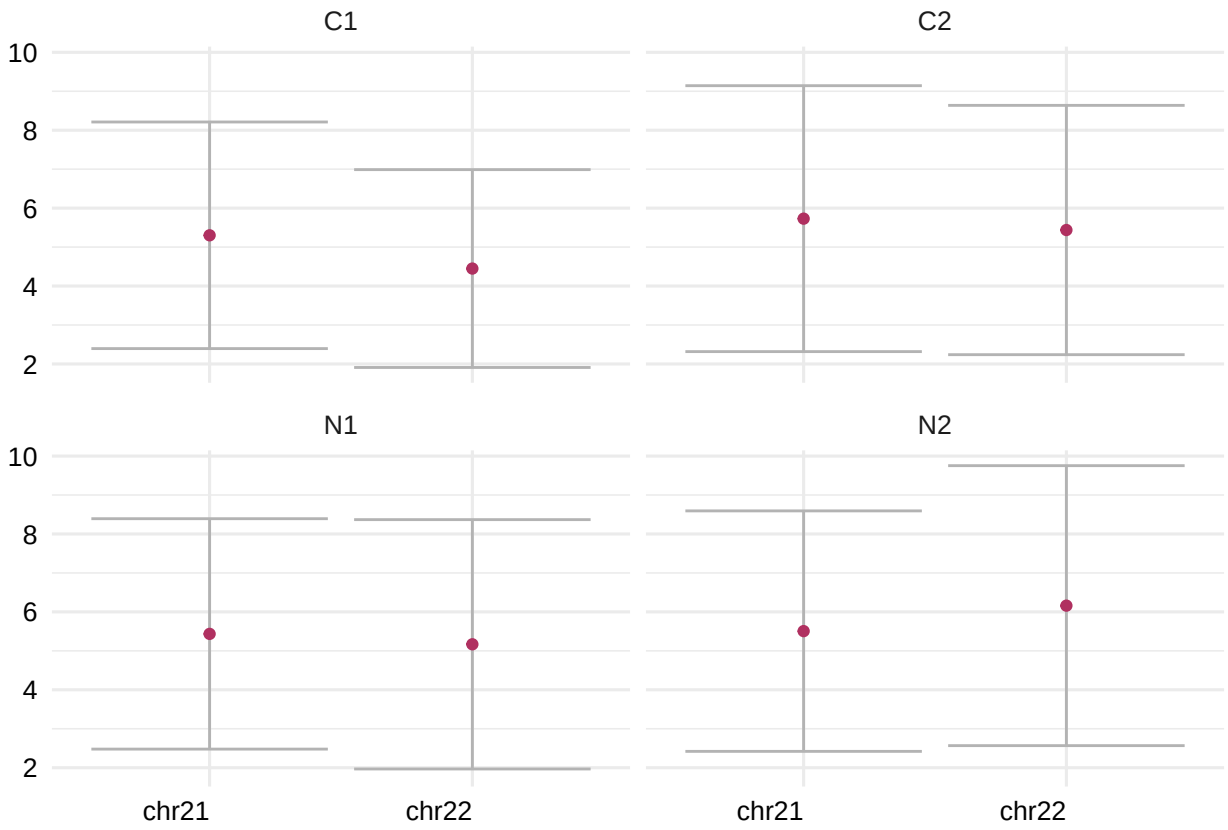
```
#>     is_h5: FALSE
#> Reference: hg19
```

Summary statsitcis

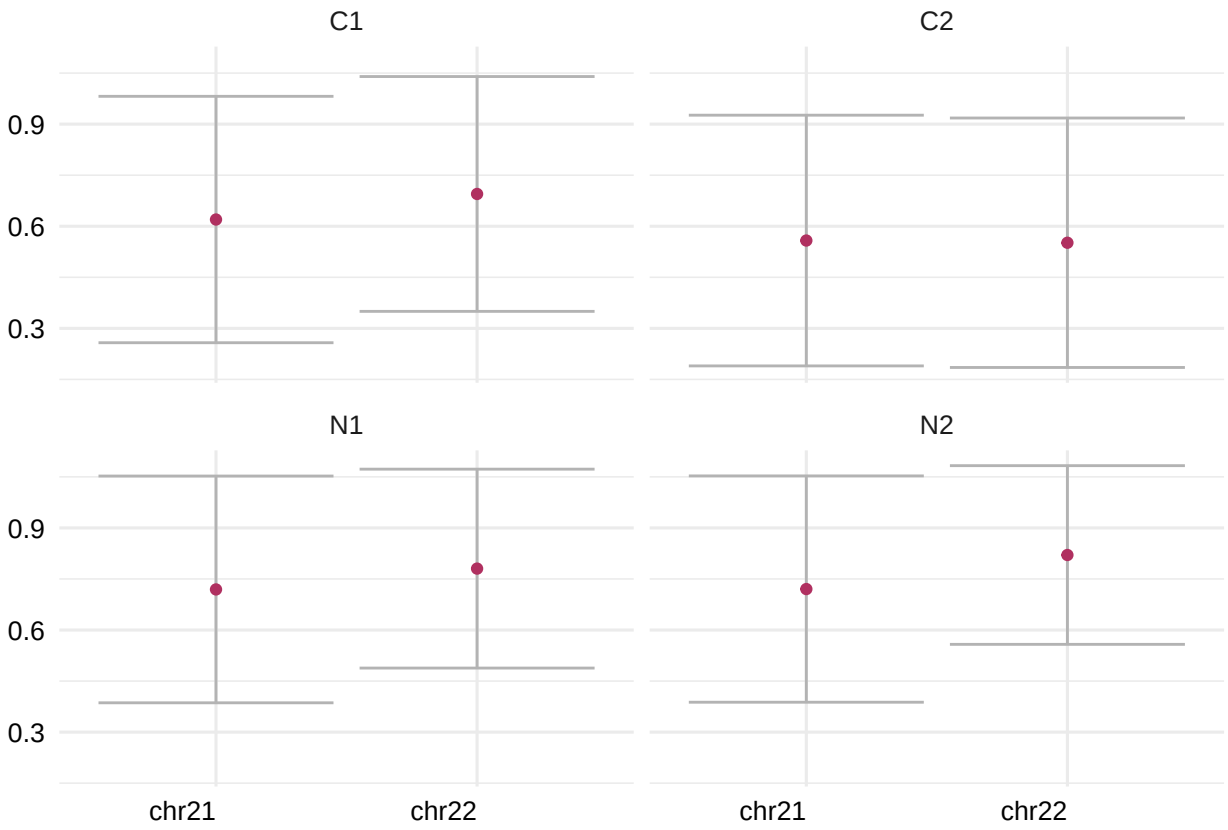
Basic summaries

```
meth_stats = get_stats(m = meth)
#> -Finished in:   0.870s elapsed (0.860s cpu)
print(meth_stats)
#>    Chromosome Sample_Name mean_meth median_meth   sd_meth mean_cov
#> 1:      chr21          C1 0.6197696   0.6666667 0.3620244 5.302564
#> 2:      chr21          C2 0.5580118         0.6 0.3683542  5.72973
#> 3:      chr21          N1 0.7192437   0.8333333 0.3331693 5.434889
#> 4:      chr21          N2 0.7203597   0.8571429 0.3325064 5.506143
#> 5:      chr22          C1 0.6948953         0.8 0.3449906 4.448413
#> 6:      chr22          C2 0.5515324   0.5857143 0.3665156 5.437984
#> 7:      chr22          N1 0.7804323   0.8888889 0.2922265 5.167969
#> 8:      chr22          N2 0.8204069           1  0.262506 6.160156
#>    median_cov   sd_cov
#> 1:          5 2.908324
#> 2:          5 3.413343
#> 3:          5 2.957009
#> 4:          5 3.086493
#> 5:          4 2.539198
#> 6:          5 3.199895
#> 7:          5 3.200354
#> 8:          5 3.593258
```

```
#Draw mean coverage per sample
plot_stats(plot_dat = meth_stats, what = "C", stat = "mean")
```



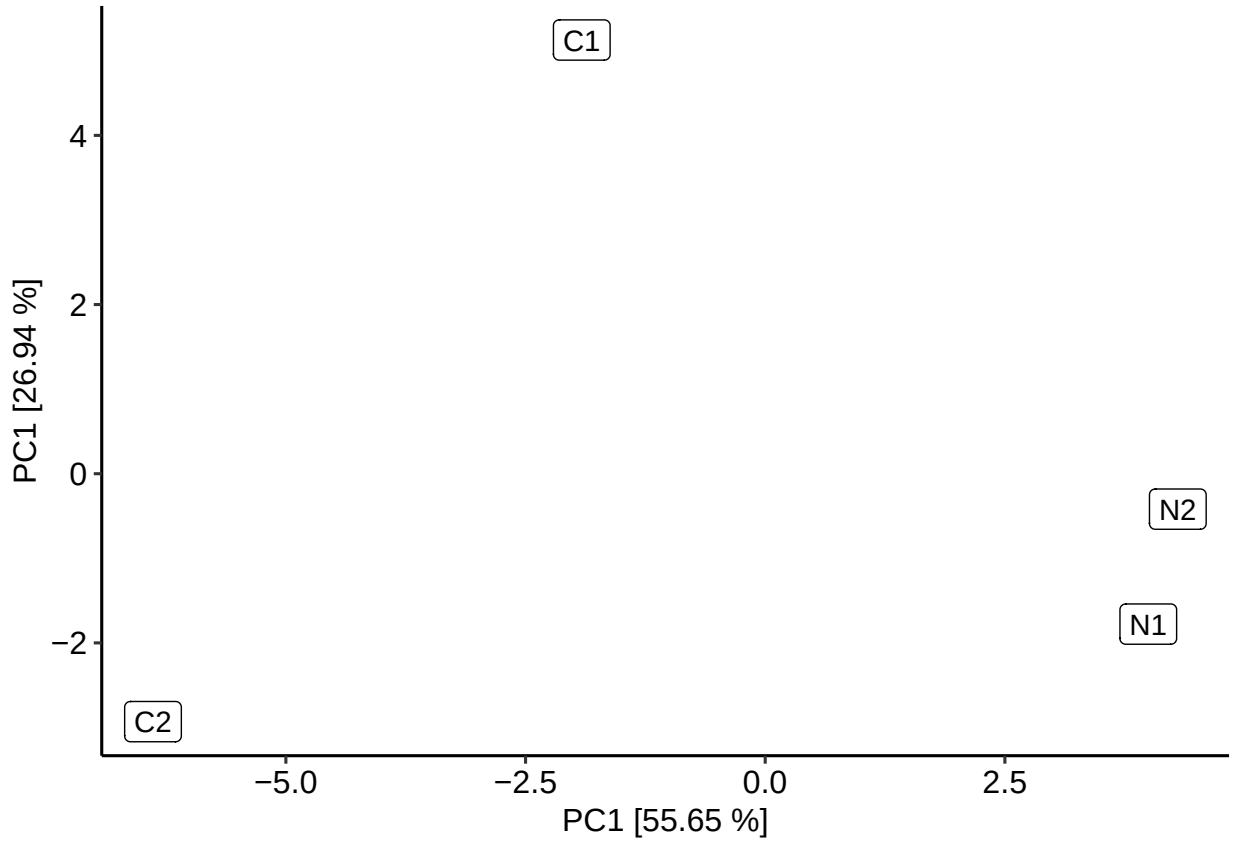

```
#Draw mean methylation per sample
plot_stats(plot_dat = meth_stats, what = "M", stat = "mean")
```



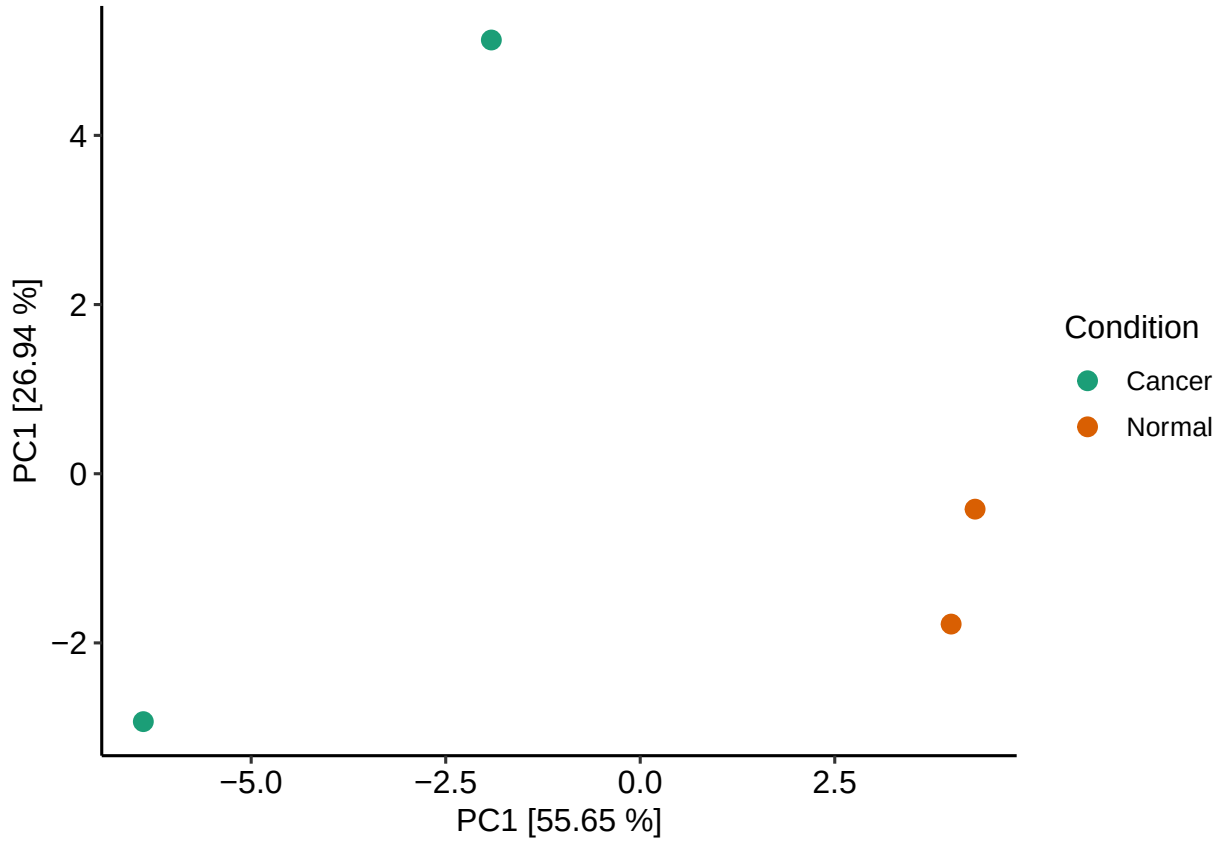
PCA

```
mpca = methrix_pca(m = meth, do_plot = FALSE)

#Plot PCA results
plot_pca(pca_res = mpca, show_labels = TRUE)
```



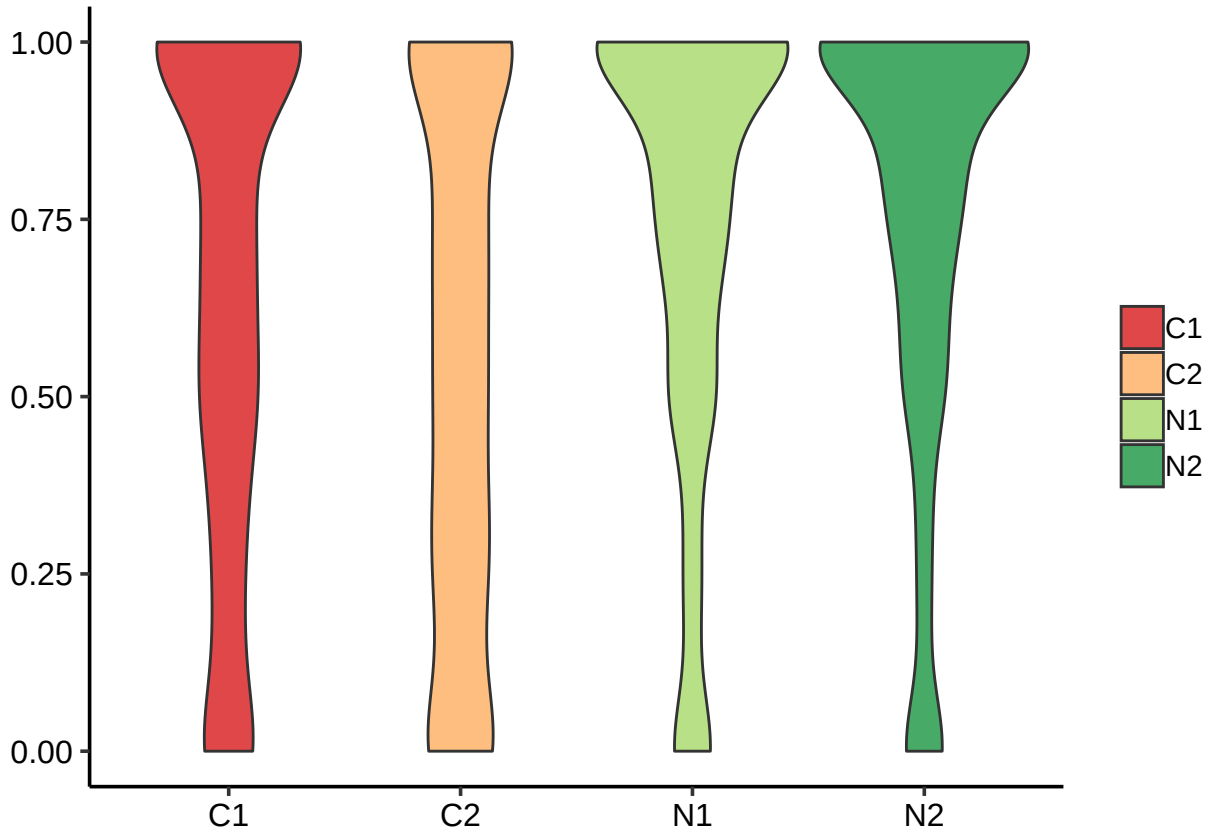

```
#Color code by an annotation
plot_pca(pca_res = mpca, m = meth, col_anno = "Condition")
```



Plotting

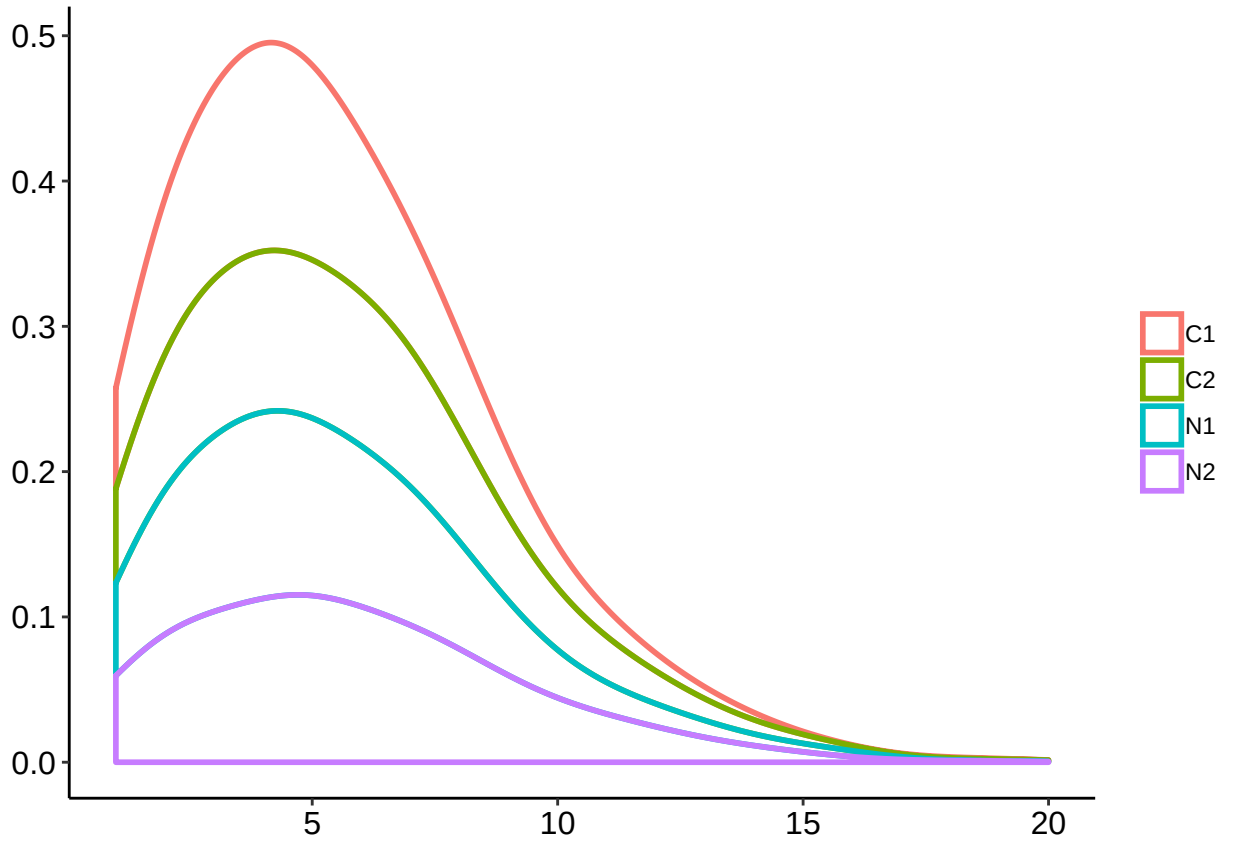
Methylation

```
#Violin plots
methrix::plot_violin(m = meth)
#> Randomly selecting 25000 sites.
#> Warning: Removed 163 rows containing non-finite values (stat_ydensity).
```



Coverage

```
methrix::plot_coverage(m = meth,type = "dens")
```



Converting methrix to BSseq

If you prefer to work with bsseq object, you can generate `bsseq` object from methrix with the `methrix2bsseq`.

```
library(bsseq)
```

```
bs_seq = methrix::methrix2bsseq(m = meth)

bs_seq
#> An object of type 'BSseq' with
#>   699 methylation loci
#>   4 samples
#> has not been smoothed
#> All assays are in-memory
```

SessionInfo

```
sessionInfo()
#> R version 3.6.0 (2019-04-26)
#> Platform: x86_64-w64-mingw32/x64 (64-bit)
#> Running under: Windows 7 x64 (build 7601) Service Pack 1
#>
```

```
#> Matrix products: default
#>
#> locale:
#> [1] LC_COLLATE=English_United States.1252
#> [2] LC_CTYPE=English_United States.1252
#> [3] LC_MONETARY=English_United States.1252
#> [4] LC_NUMERIC=C
#> [5] LC_TIME=English_United States.1252
#>
#> attached base packages:
#> [1] parallel  stats4    stats     graphics  grDevices utils     datasets
#> [8] methods   base
#>
#> other attached packages:
#>  [1] bsseq_1.20.0                    BSgenome.Hsapiens.UCSC.hg19_1.4.0
#>  [3] BSgenome_1.52.0                 rtracklayer_1.44.0
#>  [5] Biostrings_2.52.0               XVector_0.24.0
#>  [7] methrix_0.9.92                  SummarizedExperiment_1.14.0
#>  [9] DelayedArray_0.10.0             BiocParallel_1.18.0
#> [11] matrixStats_0.54.0              Biobase_2.44.0
#> [13] GenomicRanges_1.36.0            GenomeInfoDb_1.20.0
#> [15] IRanges_2.18.0                  S4Vectors_0.22.0
#> [17] BiocGenerics_0.30.0             data.table_1.12.2
#>
#> loaded via a namespace (and not attached):
#>  [1] bitops_1.0-6             fs_1.3.1
#>  [3] usethis_1.5.0            devtools_2.0.2
#>  [5] RColorBrewer_1.1-2       rprojroot_1.3-2
#>  [7] tools_3.6.0              backports_1.1.4
#>  [9] R6_2.4.0                 HDF5Array_1.12.1
#> [11] lazyeval_0.2.2           colorspace_1.4-1
#> [13] permute_0.9-5            withr_2.1.2
#> [15] tidyselect_0.2.5         prettyunits_1.0.2
#> [17] processx_3.3.1           compiler_3.6.0
#> [19] cli_1.1.0                desc_1.2.0
#> [21] labeling_0.3             scales_1.0.0
#> [23] callr_3.2.0              stringr_1.4.0
#> [25] digest_0.6.18            Rsamtools_2.0.0
#> [27] rmarkdown_1.12           R.utils_2.8.0
#> [29] pkgconfig_2.0.2          htmltools_0.3.6
#> [31] sessioninfo_1.1.1        limma_3.40.0
#> [33] rlang_0.3.4              rstudioapi_0.10
#> [35] DelayedMatrixStats_1.6.0 gtools_3.8.1
#> [37] dplyr_0.8.1              R.oo_1.22.0
#> [39] RCurl_1.95-4.12          magrittr_1.5
#> [41] GenomeInfoDbData_1.2.1   Matrix_1.2-17
#> [43] Rcpp_1.0.1               munsell_0.5.0
#> [45] Rhdf5lib_1.6.0           R.methodsS3_1.7.1
#> [47] stringi_1.4.3            yaml_2.2.0
#> [49] zlibbioc_1.30.0          rhdf5_2.28.0
#> [51] pkgbuild_1.0.3           plyr_1.8.4
#> [53] grid_3.6.0               crayon_1.3.4
#> [55] lattice_0.20-38          locfit_1.5-9.1
```

```
#> [57] knitr_1.22                 ps_1.3.0
#> [59] pillar_1.4.0               rjson_0.2.20
#> [61] pkgload_1.0.2              XML_3.98-1.20
#> [63] glue_1.3.1                 evaluate_0.13
#> [65] remotes_2.0.4              testthat_2.1.1
#> [67] gtable_0.3.0               purrr_0.3.2
#> [69] assertthat_0.2.1           ggplot2_3.1.1
#> [71] xfun_0.6                   tibble_2.1.1
#> [73] GenomicAlignments_1.20.0 memoise_1.1.0
```