

ChromENVEE: Chromatin ENVironment and Enhancer Expression

Coulée Manon

2022-10-10

Contents

Abstract	1
Citation	1
Introduction	1
Initialization of data	2
Distribution of chromatin state in the genome	3
Annotation of enhancer	4
Annotated enhancer binding to enhancer position	5
Associated gene expression to enhancer	5
Profile of enhancer annotation	6
Enhancer annotation comparison	8
Gene environment	12
Coverage of chromatin state in environment binding to TSS regions	12
Predominant state in environment binding to TSS regions	13
Session Information	14
References	16

Abstract

Citation

Introduction

ChromENVEE is a package developed to studying chromatin state without Hi-C data.

This package implements functions to associated genes with enhancers, define the chromatin environment of the gene from genomic data (e.g., ChromHMM output or a bed file). Several visualization functions are available to summarize the distribution of chromatin states, characterize genes associated with enhancers and also estimate the chromatin environment of genes.

This package is available for R version ≥ 3.6 .

```
# Loading package
library(ChromENVEE)
```

Initialization of data

`colorTable` is a data frame contains informations about chromatin state number `stateNumber`, chromatin state name `stateName` and chromatin state color `colorValue` (used for plot generation).

```
data(colorTable)
```

1	U1	TSSA	#B71C1C
2	U2	TSSFlnk	#E65100
3	U3	TSSFlnkD	#E65100
4	U4	Tx	#43A047
5	U5	TxWk	#1B5E20
6	U6	EnhG	#99FF66
7	U7	EnhG	#99FF66
8	U8	EnhA	#F5B041
9	U9	EnhWk	#FFEB3B
10	U10	ZNFRpts	#48C9B0
11	U11	Het	#B39DDB
12	U12	TssBiv	#880E4F
13	U13	EnhBiv	#666633
14	U14	ReprPC	#424949
15	U15	ReprPCWk	#7B7D7D
16	U16	Quies	#D0D3D4
17	U17	Quies	#D0D3D4

`genomeFile` is a data frame contains informations about mouse reference genome.

It is generated from bed file, in the case of this study, we used Ensembl annotation. `genomeFile` required to contains informations like chromosome (chr), gene position (start and end), strand information (strand) and gene name (gene_ENS). Score informations is suggested but not required.

```
data(genomeFile)
```

```
#>   chr  start  end strand score      gene_ENS
#> 1 chr1 3073253 3074322    +    . ENSMUSG000000102693.1
#> 2 chr1 3102016 3102125    +    . ENSMUSG000000064842.1
#> 3 chr1 3205901 3671498    -    . ENSMUSG000000051951.5
#> 4 chr1 3252757 3253236    +    . ENSMUSG000000102851.1
#> 5 chr1 3365731 3368549    -    . ENSMUSG000000103377.1
#> 6 chr1 3375556 3377788    -    . ENSMUSG000000104017.1
```

`chromatinState` is a data frame contains informations about chromatin state.

It is generated with the output of ChromHMM tools. `chromatinState` required to contains informations like chromosome (chr), genomic regions (start and end), chromatin state information (state and state_name) and sample name (name).

```
data(chromatinState)

#>      chr   start      end state name state_name
#> 1 chr10      0 3100000   U16   RS      Quies
#> 2 chr10 3100000 3109200   U11   RS      Het
#> 3 chr10 3109200 3110600   U12   RS      TssBiv
#> 4 chr10 3110600 3111000   U14   RS      ReprPC
#> 5 chr10 3111000 3111200   U13   RS      EnhBiv
#> 6 chr10 3111200 3117200   U12   RS      TssBiv
```

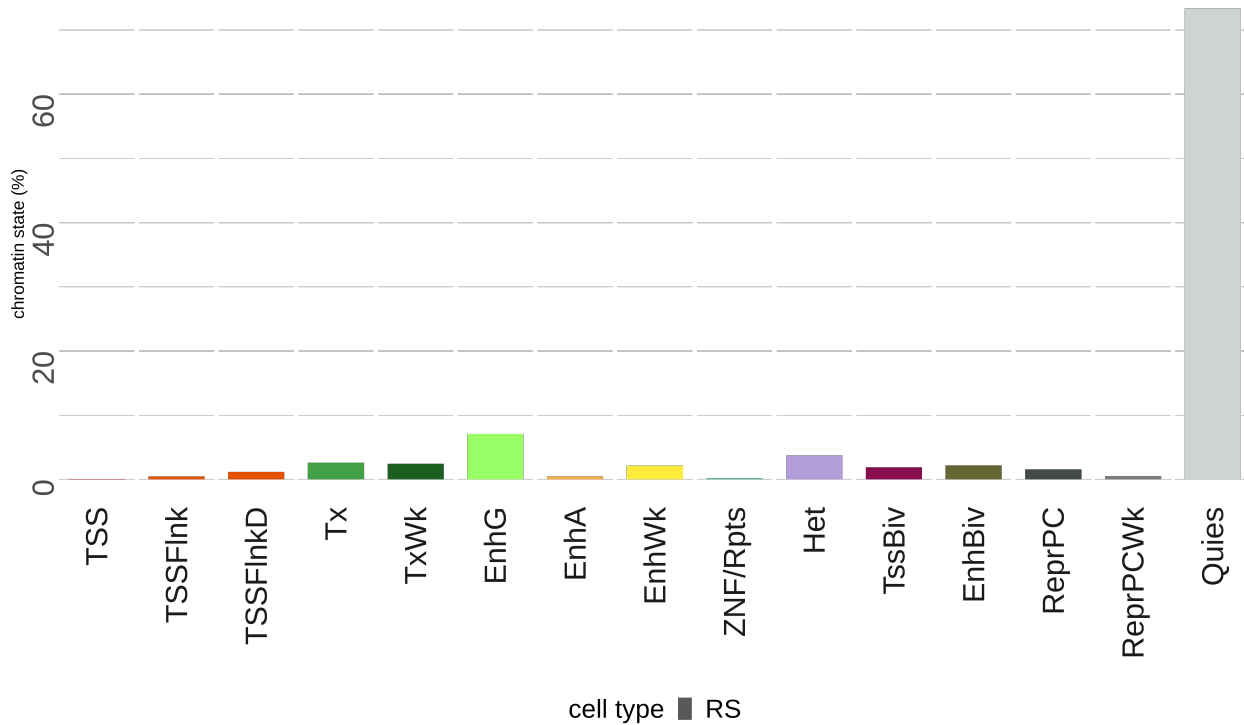
Distribution of chromatin state in the genome

We are interested to know the distribution of chromatin state in the genome.

`plotChromatinState` calculates the percentage of each chromatin state in function the length of the genome used. We obtains a data frame with the percentage of coverage for each chromatin state. It's possible to plot the result in .png file with the argument `plot = TRUE`. If you have a list of data frame, it's possible to merge all the data frame in merge data frame and in unique plot with `merge = TRUE` argument.

```
summary_chromatin_state = plotChromatinState(chromatinState, merge = TRUE, plot = FALSE,
colorTable = colorTable, filename = "")
```

```
head(summary_chromatin_state)
#>           state  coverage sample_name
#> TSSA          TSSA 0.08519426        RS
#> TSSFlnk       TSSFlnk 0.45530134        RS
#> TSSFlnkD     TSSFlnkD 1.18900667        RS
#> Tx           Tx 2.60257103         RS
#> TxWk         TxWk 2.44911129         RS
#> EnhG         EnhG 7.10081351         RS
```



Annotation of enhancer

We are interested to associated at each enhancer, genes regulated by the enhancer. We focused on enhancer chromatin state (in this study, we have 4 type of enhancer : bivalent enhancer (EnhBiv), genic enhancer (EnhG), active enhancer (EnhA) and weak enhancer (EnhWk)).

`listTableEnhancer` is a `GRanges` object or a list of `GRanges` object (produced by `GenomicRanges` package). Like `chromatinState` data frame, `listTableEnhancer` required gene information and chromatin state information. Sample name (`sample_name`) is required if you want compared enhancer annotation (see Enhancer annotation comparison).

```
data(listTableEnhancer)
```

```
#> GRanges object with 1979 ranges and 2 metadata columns:
#>      seqnames      ranges strand | chromatin_state sample_name
#>      <Rle>        <IRanges> <Rle> | <character> <character>
#> [1] chr10      9164400-9164800    * |      U13      EnhBiv
#> [2] chr10      9342200-9344000    * |      U13      EnhBiv
#> [3] chr10     10476400-10476600    * |      U13      EnhBiv
#> [4] chr10     20520200-20521000    * |      U13      EnhBiv
#> [5] chr10     20952400-20952600    * |      U13      EnhBiv
#> ...      ...      ...      ...      ...
#> [1975] chrX 144286800-144287000    * |      U13      EnhBiv
#> [1976] chrX 155128400-155129200    * |      U13      EnhBiv
#> [1977] chrX 170010800-170013800    * |      U13      EnhBiv
#> [1978] chrY      198400-198800    * |      U13      EnhBiv
#> [1979] chrY    90786000-90788000    * |      U13      EnhBiv
#> -----
#> seqinfo: 21 sequences from an unspecified genome; no seqlengths
```

Annotated enhancer binding to enhancer position

To estimate which gene is regulated by enhancer, we estimated that genes associated with enhancer, all TSS genes in interval around enhancer. `enhancerAnnotation()` uses a `GRanges` object.

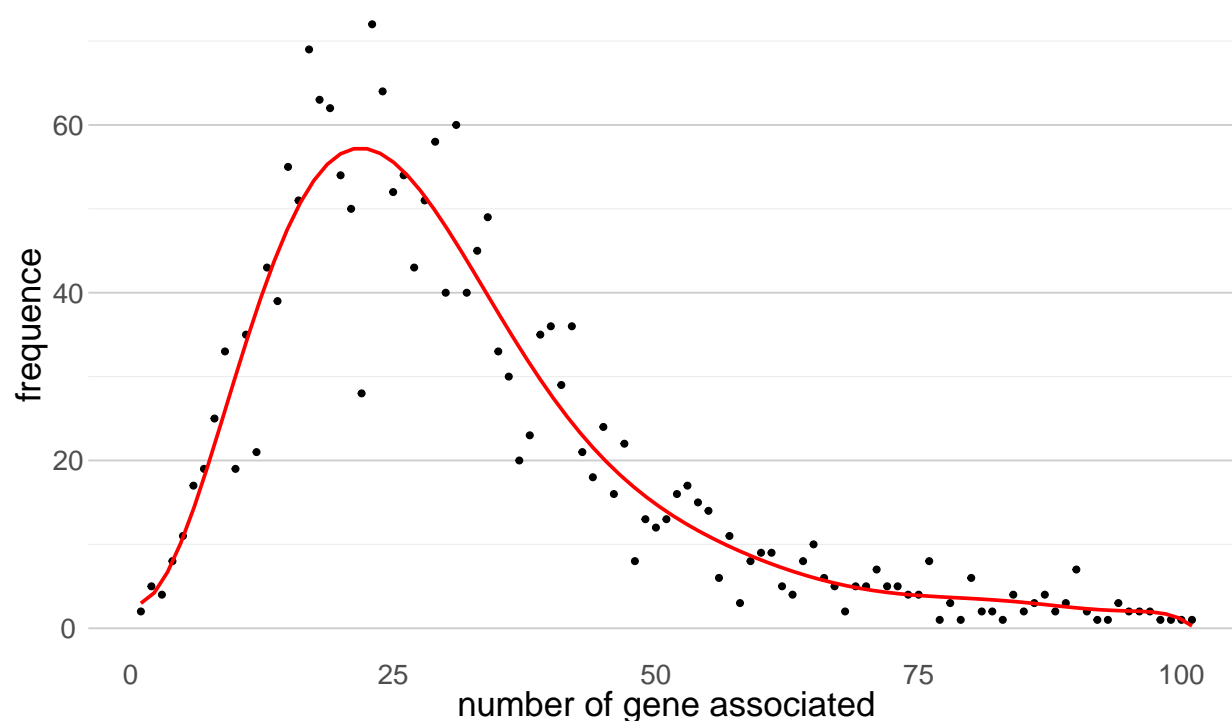
The function takes a few minutes to process in function of the length of your enhancer table. It's possible to multithread the job with the `nCore` parameter. For each enhancer position, we obtain two informations, the distance between gene and enhancer (in bp) and the gene.

```
table_enhancer_gene = enhancerAnnotation(listTableEnhancer[[1]], genome = genomeFile,  
interval = 500000, nCore = 1)
```

Number of gene associate at the enhancer

We want to know the distribution of genes associated at each enhancer using `plotGeneAssociation`.

```
plotGeneAssociation(table_enhancer_gene, all = FALSE)
```



Associated gene expression to enhancer

`geneExpression` is a data frame contains gene expression level information.

It is generated with RNAseq gene expression analysis. `geneExpression` required to contains informations like chromosome (chr), gene position (start and end), gene name (gene_ENS), strand information (strand), level of gene expression (gene_expression). Score is not required for the analysis. For gene name, you need to used the same gene name that you used to generated `genomeFile` data frame because we used the annotation to associated the couple gene-enhancer with the expression.

```
data(geneExpression)
```

```
#>           gene_ENS   chr   start       end strand score gene_expression
#> 1  ENSMUSG000000000001.4 chr3 108107280 108146146      -      .      27.7106904
#> 2  ENSMUSG0000000000028.15 chr16 18780447 18811987      -      .      23.5842993
#> 3  ENSMUSG0000000000031.16 chr7 142575529 142578143      -      .       0.9386427
#> 4  ENSMUSG0000000000037.16 chrX 161117193 161258213      +      .      14.4548991
#> 5  ENSMUSG0000000000049.11 chr11 108343354 108414396      +      .      36.6169129
#> 6  ENSMUSG0000000000056.7 chr11 121237253 121255856      +      .       5.2791187
```

We associated the level of gene expression at each gene-enhancer couple estimate with `enhancerAnnotation` function.

According to `geneExpression` data frame, it's possible that gene-enhancer couple has not expression level, in this case, we obtains NA value.

```
table_enhancer_gene_expression = enhancerExpression(table_enhancer_gene,
geneExpressionTable = geneExpression)
```

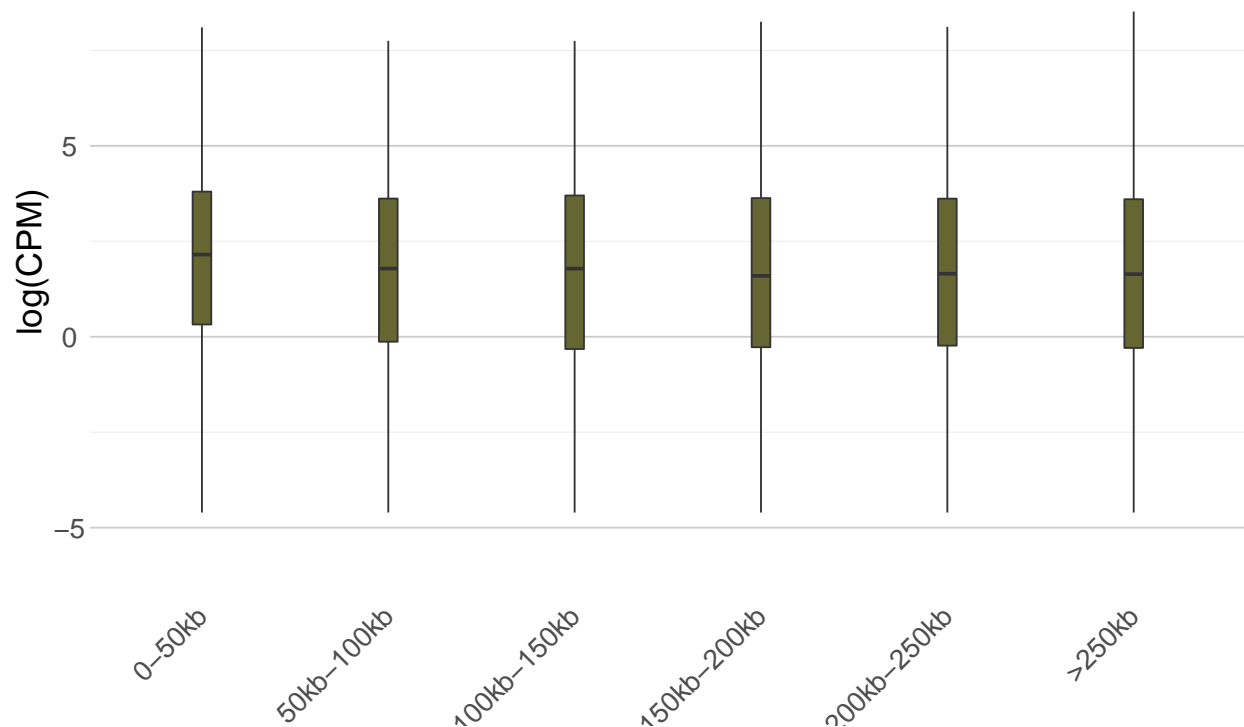
```
#> GRanges object with 6 ranges and 8 metadata columns:
#>      seqnames          ranges strand | chromatin_state sample_name start_500kb
#>      <Rle>          <IRanges> <Rle> | <character> <character> <numeric>
#> 1   chr10   9164400-9164800      * |          U13      EnhBiv      8664400
#> 1   chr10   9342200-9344000      * |          U13      EnhBiv      8842200
#> 1   chr10 10476400-10476600      * |          U13      EnhBiv      9976400
#> 1   chr10 20520200-20521000      * |          U13      EnhBiv     20020200
#> 1   chr10 20952400-20952600      * |          U13      EnhBiv     20452400
#> 1   chr10 21309400-21310600      * |          U13      EnhBiv     20809400
#>      end_500kb gene_association          distance          gene_list
#>      <numeric>      <integer>      <character>      <character>
#> 1   9664800              19 451159;278330;340253.. ENSMUSG000000111215.1..
#> 1   9844000              21 456130;480757;457563.. ENSMUSG000000015305.6..
#> 1  10976600              20 499773;435480;392457.. ENSMUSG000000101621.2..
#> 1  21021000              16 371729;318362;311710.. ENSMUSG000000019996.1..
#> 1  21452600              21 227322;432632;326765.. ENSMUSG000000019990.1..
#> 1  21810600              21 430427;356853;275607.. ENSMUSG000000111177.1..
#>      gene_expression
#>      <character>
#> 1 NA;12.8456863815602;..
#> 1 12.8456863815602;2.0..
#> 1 NA;NA;NA;NA;NA;NA;NA..
#> 1 102.374504394998;2.0..
#> 1 0.571438637996035;3...
#> 1 NA;399.268224715743;..
#> -----
#> seqinfo: 21 sequences from an unspecified genome; no seqlengths
```

Profile of enhancer annotation

Distance gene-enhancer according to their expression

We generated plot to estimated the level of gene expression according to the distance between gene and enhancer using `plotDistanceExpression`. The distance is estimated with `limit` argument and clustered in 6 distance groups like the following plot.

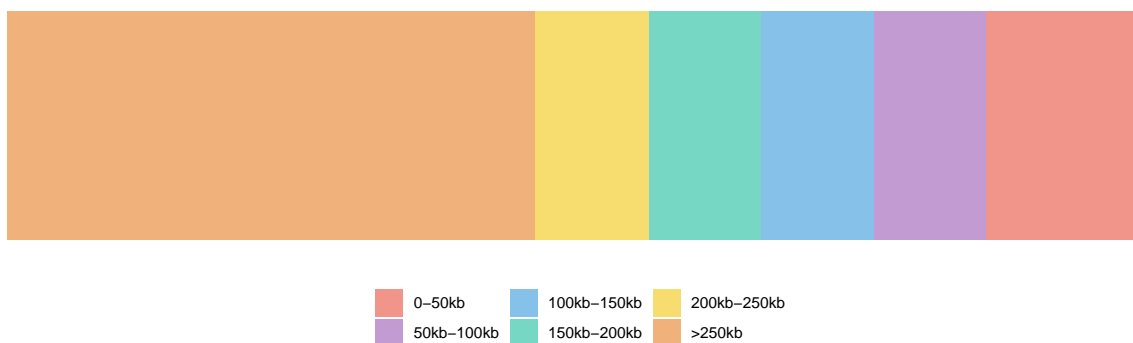
```
plotDistanceExpression(table_enhancer_gene_expression, colorTable = colorTable,
limit = 500000)
```



Distance gene-enhancer

We generated plot to estimated the distribution of gene according to the distance between gene and enhancer using `plotGeneDistance`. The distance is estimated with `limit` argument and clusterized in 6 distance groups like the following plot.

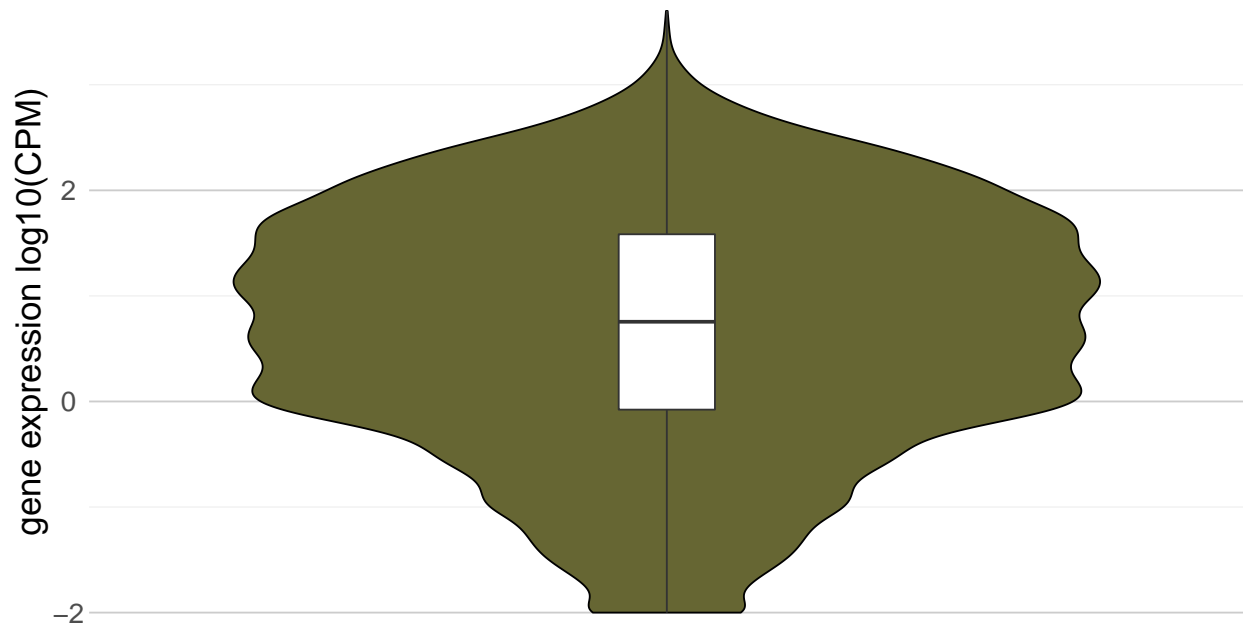
```
plotGeneDistance(table_enhancer_gene_expression, limit = 500000, xlab = "",
ylab = "distance enhancer-gene (bp)")
```



Enhancer expression

We generated plot with the distribution of gene expression associated at enhancer region using `plotEnhancerExpression`. It's possible to rescale plot with `scale` argument ('none', 'log10' and 'log2' are accepted).

```
plotEnhancerExpression(table_enhancer_gene_expression, scale = "log10",
colorTable = colorTable, ylab = "gene expression log10(CPM)")
```



Enhancer annotation comparison

It's possible to compare different categories of enhancer. For that it's necessary to use a list of GRanges objects each containing data like `listTableEnhancer` data. Contrary to individual analysis, each GRanges object in the list requires sample information (`sample_name`).

The first step is associating a gene to each enhancer using `enhancerAnnotation()` on the list of enhancer. After the gene association, we associated the gene expression using `enhancerExpression()`. In the case of this study, all enhancer categories come from the same cell type, we also used the same `geneExpression` data frame.

```
list_table_enhancer_gene = lapply(listTableEnhancer, enhancerAnnotation,
genome = genomeFile, interval = 500000, nCore = 1)
listTableEnhancerGeneExpression = lapply(list_table_enhancer_gene, enhancerExpression,
geneExpressionTable = geneExpression)
```

This process takes many times.

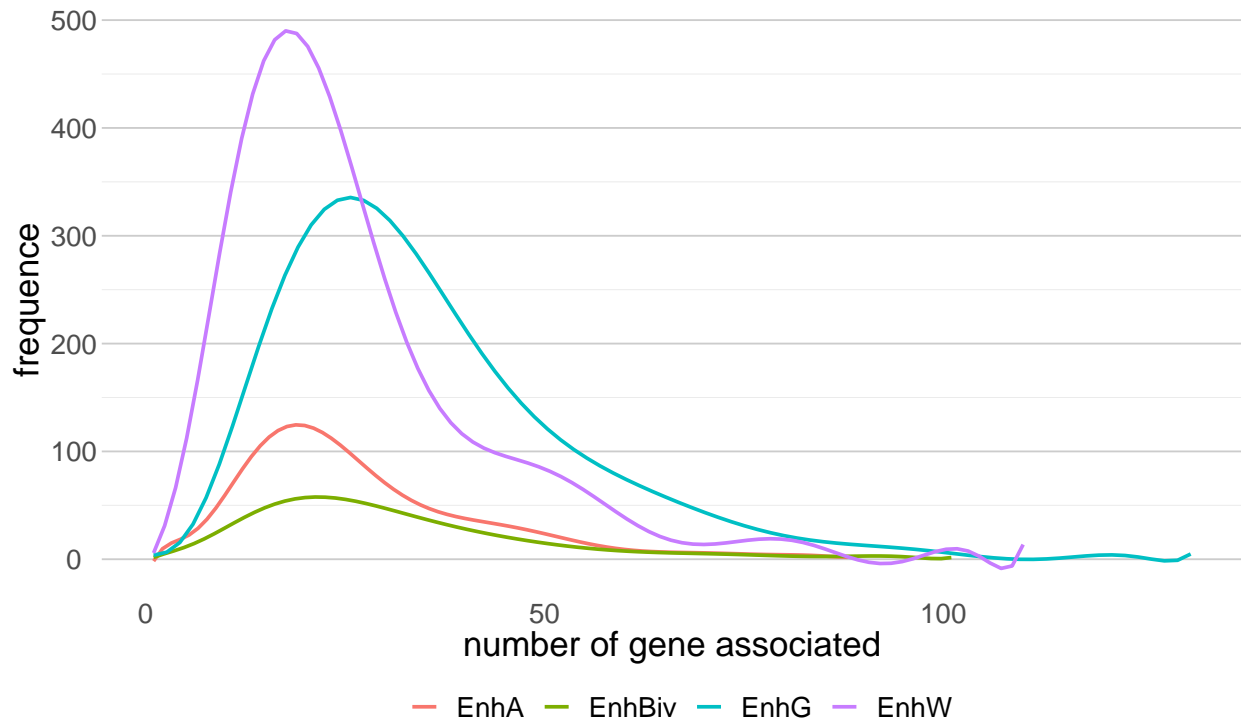
To reduce time, you can load `listTableEnhancerGeneExpression` data to process the next analysis.

```
data(listTableEnhancerGeneExpression)
```

Number of gene associated at the enhancer

We want to know the distribution of genes associated at each enhancer using `plotGeneAssociation`. `all = TRUE` parameter is used to compile all enhancer tables in the same file.

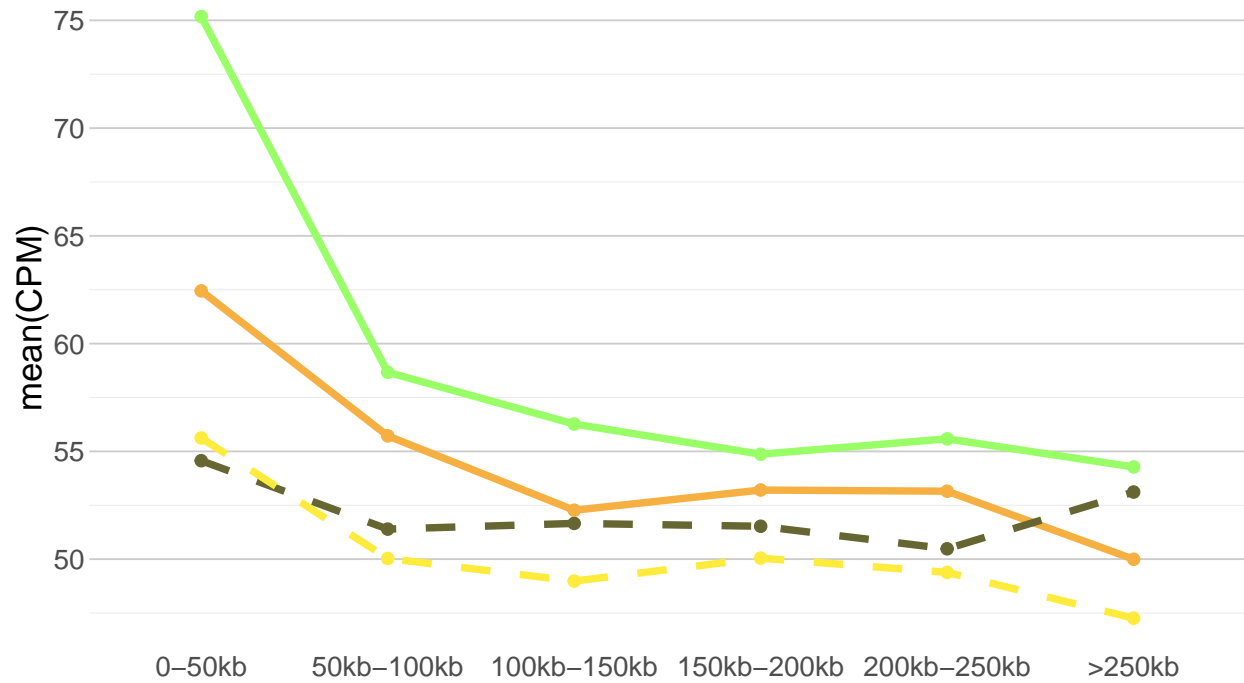

```
plotGeneAssociation(listTableEnhancerGeneExpression, all = TRUE)
```



Distance gene-enhancer according to their expression

We generate plot to estimated the level of gene expression according to the distance between gene and enhancer using `plotDistanceExpression`. The distance is estimated with `limit` argument and clustered in 6 distance groups like the following plot. In the case of list analysis, the function showed the mean of expression.

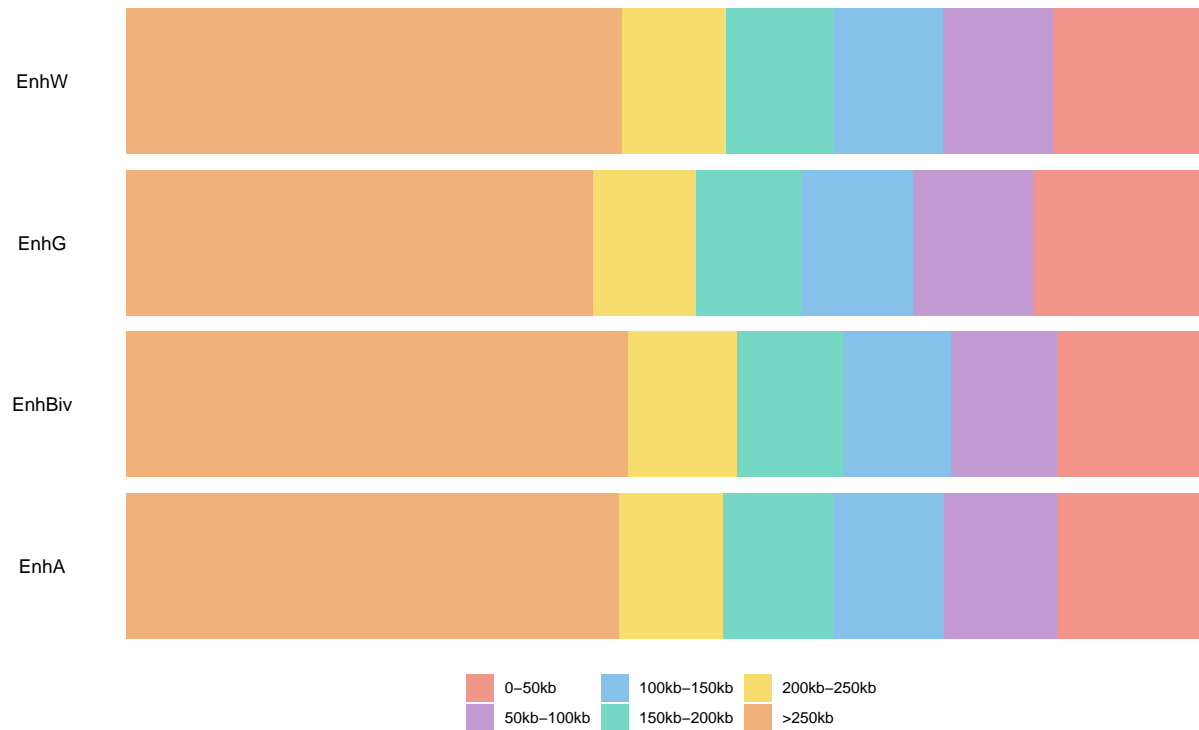
```
plotDistanceExpression(listTableEnhancerGeneExpression, colorTable = colorTable,
limit = 500000)
```



Distance gene-enhancer

We generate plot to estimated the distribution of gene according to the distance between gene and enhancer using `plotGeneDistance`. The distance is estimated with `limit` argument and clusterized in 6 distance groups like the following plot.

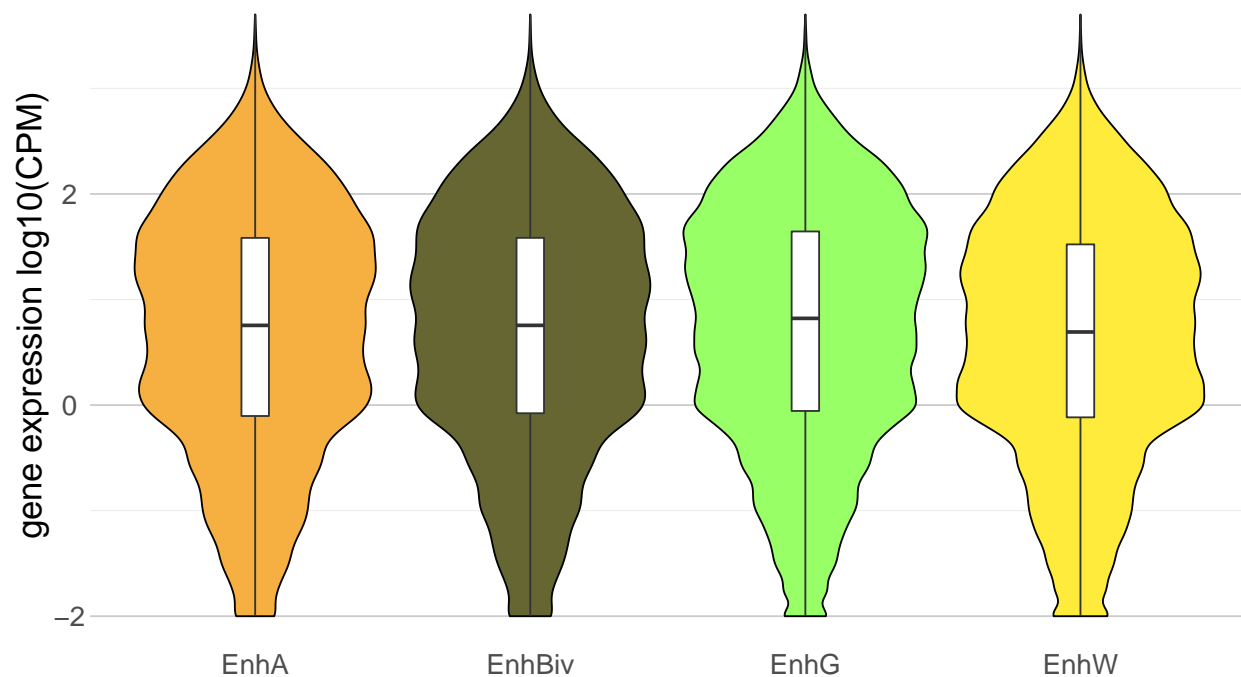
```
plotGeneDistance(listTableEnhancerGeneExpression, limit = 500000,
xlab = "", ylab = "distance enhancer-gene (bp)")
```



Enhancer expression

We generate plot with the distribution of gene expression associated at enhancer region using `plotEnhancerExpression`. It's possible to rescale plot with `scale` argument ('none', 'log10' and 'log2' are accepted).

```
plotEnhancerExpression(listTableEnhancerGeneExpression, scale = "log10",
colorTable = colorTable, ylab = "gene expression log10(CPM)")
```



Gene environment

We are interested to studying the chromatin environment of gene. For that we need gene information data (`geneExpression`) and chromatin state data (`chromatinState`).

```
data(geneExpression)
data(chromatinState)
```

Coverage of chromatin state in environment binding to TSS regions

`geneEnvironment` is a function able to estimate the chromatin state environment of gene. For that, we estimated the environment size around gene TSS with `interval` parameter. For each gene, we obtains informations about the coverage of each chromatin state (`state` parameter) in the environment.

`geneEnvironment` may take few minutes in function the number of genes analyzed.

```
table_overlapping = geneEnvironment(geneExpression, chromatinState,
stateOrder = unique(colorTable$stateName), interval = 3000)
rownames(table_overlapping) = table_overlapping$gene_ENS
```

```
#>
#> ENSMUSG000000000001.4 ENSMUSG000000000001.4 chr3 108107280 108146146 -
#> ENSMUSG000000000028.15 ENSMUSG000000000028.15 chr16 18780447 18811987 -
#> ENSMUSG000000000031.16 ENSMUSG000000000031.16 chr7 142575529 142578143 -
#> ENSMUSG000000000037.16 ENSMUSG000000000037.16 chrX 161117193 161258213 +
#> ENSMUSG000000000049.11 ENSMUSG000000000049.11 chr11 108343354 108414396 +
#> ENSMUSG000000000056.7 ENSMUSG000000000056.7 chr11 121237253 121255856 +
#>
#> score gene_expression TSS TSS_moins_3kb
```

```

#> ENSMUSG000000000001.4      .      27.7106904 108146146      108143146
#> ENSMUSG000000000028.15     .      23.5842993 18811987      18808987
#> ENSMUSG000000000031.16     .       0.9386427 142578143      142575143
#> ENSMUSG000000000037.16     .      14.4548991 161117193      161114193
#> ENSMUSG000000000049.11     .      36.6169129 108343354      108340354
#> ENSMUSG000000000056.7      .       5.2791187 121237253      121234253
#>
#>      TSS_plus_3kb      TSSA      TSSFlnk TSSFlnkD Tx TxWk
#> ENSMUSG000000000001.4      108149146 0.00000000 0.00000000      0 0 0
#> ENSMUSG000000000028.15      18814987 0.00000000 0.06666667      0 0 0
#> ENSMUSG000000000031.16      142581143 0.00000000 0.00000000      0 0 0
#> ENSMUSG000000000037.16      161120193 0.03333333 0.40000000      0 0 0
#> ENSMUSG000000000049.11      108346354 0.00000000 0.00000000      0 0 0
#> ENSMUSG000000000056.7      121240253 0.00000000 0.06666667      0 0 0
#>
#>      EnhG EnhA  EnhWk ZNFRpts Het      TssBiv EnhBiv ReprPC
#> ENSMUSG000000000001.4 0.7423333 0 0.0000      0 0 0.2576667      0.0 0.0000
#> ENSMUSG000000000028.15 0.6333333 0 0.0000      0 0 0.3000000      0.0 0.0000
#> ENSMUSG000000000031.16 0.0000000 0 0.0000      0 0 0.0000000      0.4 0.3095
#> ENSMUSG000000000037.16 0.0000000 0 0.1655      0 0 0.0000000      0.0 0.0000
#> ENSMUSG000000000049.11 0.0000000 0 0.0000      0 0 0.3000000      0.3 0.3410
#> ENSMUSG000000000056.7 0.6000000 0 0.0000      0 0 0.3333333      0.0 0.0000
#>
#>      ReprPCWk      Quies
#> ENSMUSG000000000001.4      0 0.0000000
#> ENSMUSG000000000028.15      0 0.0000000
#> ENSMUSG000000000031.16      0 0.2905000
#> ENSMUSG000000000037.16      0 0.4011667
#> ENSMUSG000000000049.11      0 0.0590000
#> ENSMUSG000000000056.7      0 0.0000000

```

Predominant state in environment binding to TSS regions

`predominantState` able to estimated the predominant chromatin state in the environment of the gene. The function estimate as predominant the chromatin state with the higher coverage in the environment. Genes are clusterized in function their chromatin state using `umap` package. The function return a data frame with information about the predominant chromatin state and UMAP dimension.

```

result_umap = predominantState(table_overlapping, state = unique(colorTable$stateName),
header = unique(colorTable$stateName), neighbors = 32, metric = "euclidean", dist = 0.5)
#>
#> ==> It will be take few minutes to process

```

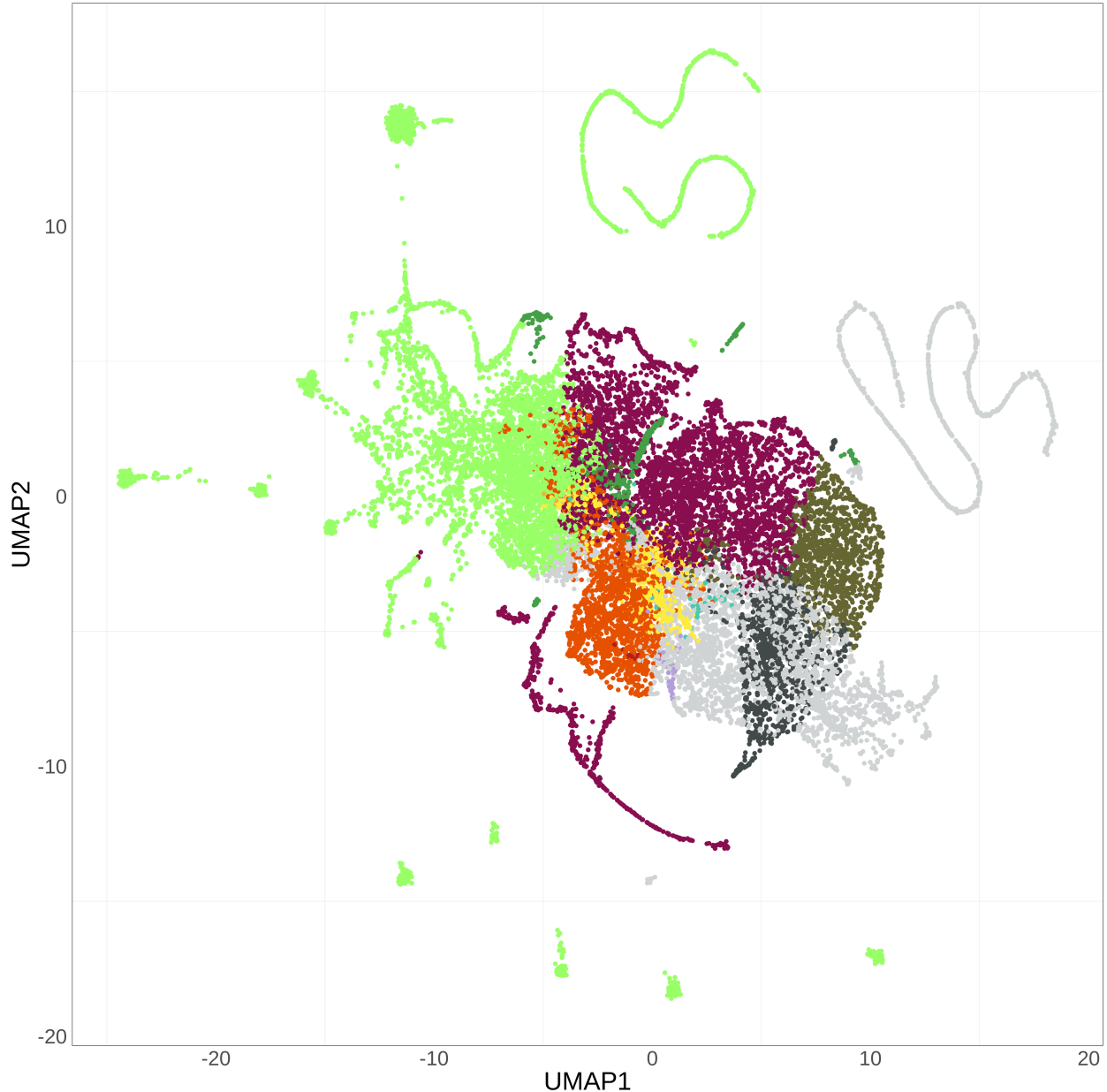
```

#>
#>      TSSA      TSSFlnk TSSFlnkD Tx TxWk      EnhG EnhA
#> ENSMUSG000000000001.4 0.00000000 0.00000000      0 0 0 0.7423333 0
#> ENSMUSG000000000028.15 0.00000000 0.06666667      0 0 0 0.6333333 0
#> ENSMUSG000000000031.16 0.00000000 0.00000000      0 0 0 0.0000000 0
#> ENSMUSG000000000037.16 0.03333333 0.40000000      0 0 0 0.0000000 0
#> ENSMUSG000000000049.11 0.00000000 0.00000000      0 0 0 0.0000000 0
#> ENSMUSG000000000056.7 0.00000000 0.06666667      0 0 0 0.6000000 0
#>
#>      EnhWk ZNFRpts Het      TssBiv EnhBiv ReprPC ReprPCWk
#> ENSMUSG000000000001.4 0.0000      0 0 0.2576667      0.0 0.0000      0
#> ENSMUSG000000000028.15 0.0000      0 0 0.3000000      0.0 0.0000      0
#> ENSMUSG000000000031.16 0.0000      0 0 0.0000000      0.4 0.3095      0
#> ENSMUSG000000000037.16 0.1655      0 0 0.0000000      0.0 0.0000      0
#> ENSMUSG000000000049.11 0.0000      0 0 0.3000000      0.3 0.3410      0
#> ENSMUSG000000000056.7 0.0000      0 0 0.3333333      0.0 0.0000      0

```

```
#>
#> ENSMUSG000000000001.4 0.0000000 -15.3955897 -12.093308 EnhG
#> ENSMUSG000000000028.15 0.0000000 3.1386058 11.433336 EnhG
#> ENSMUSG000000000031.16 0.2905000 -0.1952710 -8.802595 EnhBiv
#> ENSMUSG000000000037.16 0.4011667 -4.7806753 -2.087036 Quies
#> ENSMUSG000000000049.11 0.0590000 -0.9630213 -5.769000 ReprPC
#> ENSMUSG000000000056.7 0.0000000 2.4736888 11.123211 EnhG
```

It's an example of UMAP representation to visualized the predominant chromatin state in each gene.



Session Information

Here is the output of `sessionInfo()` on the system on which this document was compiled:

```

#> R version 4.1.3 (2022-03-10)
#> Platform: x86_64-conda-linux-gnu (64-bit)
#> Running under: Ubuntu 18.04.6 LTS
#>
#> Matrix products: default
#> BLAS/LAPACK: /home/mcoulee/anaconda3/envs/R_package_3/lib/libopenblas-r0.3.20.so
#>
#> locale:
#>  [1] LC_CTYPE=fr_FR.UTF-8      LC_NUMERIC=C
#>  [3] LC_TIME=fr_FR.UTF-8      LC_COLLATE=fr_FR.UTF-8
#>  [5] LC_MONETARY=fr_FR.UTF-8  LC_MESSAGES=fr_FR.UTF-8
#>  [7] LC_PAPER=fr_FR.UTF-8     LC_NAME=C
#>  [9] LC_ADDRESS=C             LC_TELEPHONE=C
#> [11] LC_MEASUREMENT=fr_FR.UTF-8 LC_IDENTIFICATION=C
#>
#> attached base packages:
#> [1] grid      stats      graphics  grDevices  utils      datasets  methods
#> [8] base
#>
#> other attached packages:
#> [1] gridExtra_2.3    ChromENVEE_1.1.7
#>
#> loaded via a namespace (and not attached):
#>  [1] Rcpp_1.0.9          lattice_0.20-45      png_0.1-7
#>  [4] prettyunits_1.1.1   ps_1.7.1            assertthat_0.2.1
#>  [7] rprojroot_2.0.3     digest_0.6.29       utf8_1.2.2
#> [10] RSpectra_0.16-1     R6_2.5.1            GenomeInfoDb_1.30.1
#> [13] stats4_4.1.3        evaluate_0.15       highr_0.9
#> [16] ggplot2_3.3.6       pillar_1.8.1        zlibbioc_1.40.0
#> [19] rlang_1.0.5         callr_3.7.1         S4Vectors_0.32.4
#> [22] Matrix_1.4-1        reticulate_1.26     rmarkdown_2.14
#> [25] labeling_0.4.2      splines_4.1.3       devtools_2.4.3
#> [28] stringr_1.4.1       RCurl_1.98-1.8      munsell_0.5.0
#> [31] umap_0.2.9.0        compiler_4.1.3      xfun_0.31
#> [34] askpass_1.1         pkgconfig_2.0.3     BiocGenerics_0.40.0
#> [37] pkgbuild_1.3.1      mgcv_1.8-40         htmltools_0.5.3
#> [40] openssl_2.0.3       tidyselect_1.1.2    tibble_3.1.8
#> [43] GenomeInfoDbData_1.2.7 IRanges_2.28.0      fansi_1.0.3
#> [46] crayon_1.5.1        dplyr_1.0.9         withr_2.5.0
#> [49] bitops_1.0-7        nlme_3.1-158        jsonlite_1.8.0
#> [52] gtable_0.3.1        lifecycle_1.0.2     DBI_1.1.3
#> [55] magrittr_2.0.3      scales_1.2.1        cli_3.4.0
#> [58] stringi_1.7.8       cachem_1.0.6        farver_2.1.1
#> [61] XVector_0.34.0      fs_1.5.2            remotes_2.4.2
#> [64] ellipsis_0.3.2      vctrs_0.4.1         generics_0.1.3
#> [67] tools_4.1.3         glue_1.6.2          purrr_0.3.4
#> [70] processx_3.7.0      pkgload_1.3.0       parallel_4.1.3
#> [73] fastmap_1.1.0       yaml_2.3.5          colorspace_2.0-3
#> [76] GenomicRanges_1.46.1 sessioninfo_1.2.2   memoise_2.0.1
#> [79] knitr_1.39          usethis_2.1.6

```

References

Ernst J, Kellis M. ChromHMM: automating chromatin-state discovery and characterization. *Nature Methods*, 9:215-216, 2012

Papier scientifique associé

McInnes, Leland, and John Healy. “UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction.” *arXiv:1802.03426*.

Lawrence M, Huber W, Pagès H, Aboyoun P, Carlson M, Gentleman R, Morgan M, Carey V (2013). “Software for Computing and Annotating Genomic Ranges.” *PLoS Computational Biology*, 9. doi: 10.1371/journal.pcbi.1003118, <http://www.ploscompbiol.org/article/info%3Adoi%2F10.1371%2Fjournal.pcbi.1003118>.