ChromENVEE: Chromatin ENVironment and Enhancer-dependent Expression

Manon Coulée

2022-12-12

Contents

Abstract

Standard analyses on ChIPseq data provide information (annotation, enrichment level) at the gene body level but do not necessarily investigate other genomic regions. ChromHMM R package allows to go further by predicting chromatin states using ChIPSeq datasets for several histone marks. The present R package ChromENVEE uses the chromatin states obtained by ChromHMM and compare them with transcriptomic data (RNAseq) and other ChIP-Seq data.

Specifically, ChromENVEE implements functions to associate all the neighbouring genes to a list of enhancers and to define the chromatin environment of genes using chromatin states informations. Several visualization functions are available to summarize the distribution of chromatin states, characterize genes associated with enhancers and also assign chromatin environment to genes.

Citation

If you use ChromENVEE in published research, please cite:

- Manon Coulee, Guillaume Meurice, Julie Cocquet* and Laila El Khattabi* (2022). ChromENVEE: Chromatin Environment and Enhancer-dependent Expression. R package version 1.1.8. *co-authorship

Introduction

ChromENVEE (**C**hromatin **ENV**ironment and **E**nhancer-dependent **E**xpression) is a package that was developed to define chromatin dynamics in a specific cell type and to characterize a histone mark at the enhancer level and its chromatin environment.

ChromHMM (*Ersnt et al, 2012*) is a tool using the Hidden Markov Model (HMM) method to predict the most likely chromatin state of each genomic segment. The tool uses ChIPseq data from multiple epigenetic marks to predict chromatin states, each characterized by at least one epigenetic mark. In the case of this present study, six epigenetic marks from 15 different cell types were used to build a model of 18 chromatin states.

The package contains several applications all using the results obtained with ChromHMM tools.

- It characterizes the distribution of the chromatin states in a given cell type.
- The package can associate chromatin states defined as enhancers with genes located nearby.
- Using transcriptomic (RNAseq) data it can also analyze the expression of those nearby genes and produce graphs to visualize the results. ChomENVEE can also determine the chromatin environment of a gene and estimate the predominant chromatin state.

The package was developed to in depth characterize a chromatin mark and correlate it with gene expression and chromatin environment in given cell types. In the present study, we focused on the chromatin mark H3K79me2 because two recent studies had shown that the presence of H3K79me2 at a subset of active enhancers can regulate gene expression (*Ferrari et al, 2020*; *Godfrey et al, 2019*).

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

Data initialization

`colorTable` is a dataframe that gives the following information: chromatin state numbers (`stateNumber`), chromatin state names (`stateName`) and chromatin state colors (`colorValue`). This table is necessary for plot generation. `colorValue` accepts as value hex code and/or color name code.

```
data(colorTable)
```

	stateNumber	stateName	colorValue
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
18	U18	Quies	#D0D3D4

`genomeFile` is a GRanges object generated from an annotation bed file. In the case of this present study, we used the mouse Ensembl annotation file.

`genomeFile` should contain the following information: chromosome (chr), gene position (start and end), strand information (strand) and gene name (gene_ENS). The score information is suggested but not mandatory.

```
data(genomeFile)
```

```
#> GRanges object with 6 ranges and 2 metadata columns:
#>       seqnames          ranges strand |       score             gene_ENS
#>          <Rle>       <IRanges>  <Rle> | <character>          <character>
#>   [1]     chr1 3073253-3074322      + |           . ENSMUSG00000102693.1
#>   [2]     chr1 3102016-3102125      + |           . ENSMUSG00000064842.1
#>   [3]     chr1 3205901-3671498      - |           . ENSMUSG00000051951.5
#>   [4]     chr1 3252757-3253236      + |           . ENSMUSG00000102851.1
#>   [5]     chr1 3365731-3368549      - |           . ENSMUSG00000103377.1
#>   [6]     chr1 3375556-3377788      - |           . ENSMUSG00000104017.1
#>   -------
#>   seqinfo: 22 sequences from mm10 genome; no seqlengths
```

`chromatinState` is a GRanges object that contains chromatin states information. It is generated with the output of the ChromHMM tool.

`chromatinState` should contain the following information: chromosome (chr), genomic regions (start and end), chromatin states (state and state_name) and sample name (name).

```
data(chromatinState)
```

```
#> GRanges object with 6 ranges and 3 metadata columns:
#>       seqnames          ranges strand |       state        name state_name
```

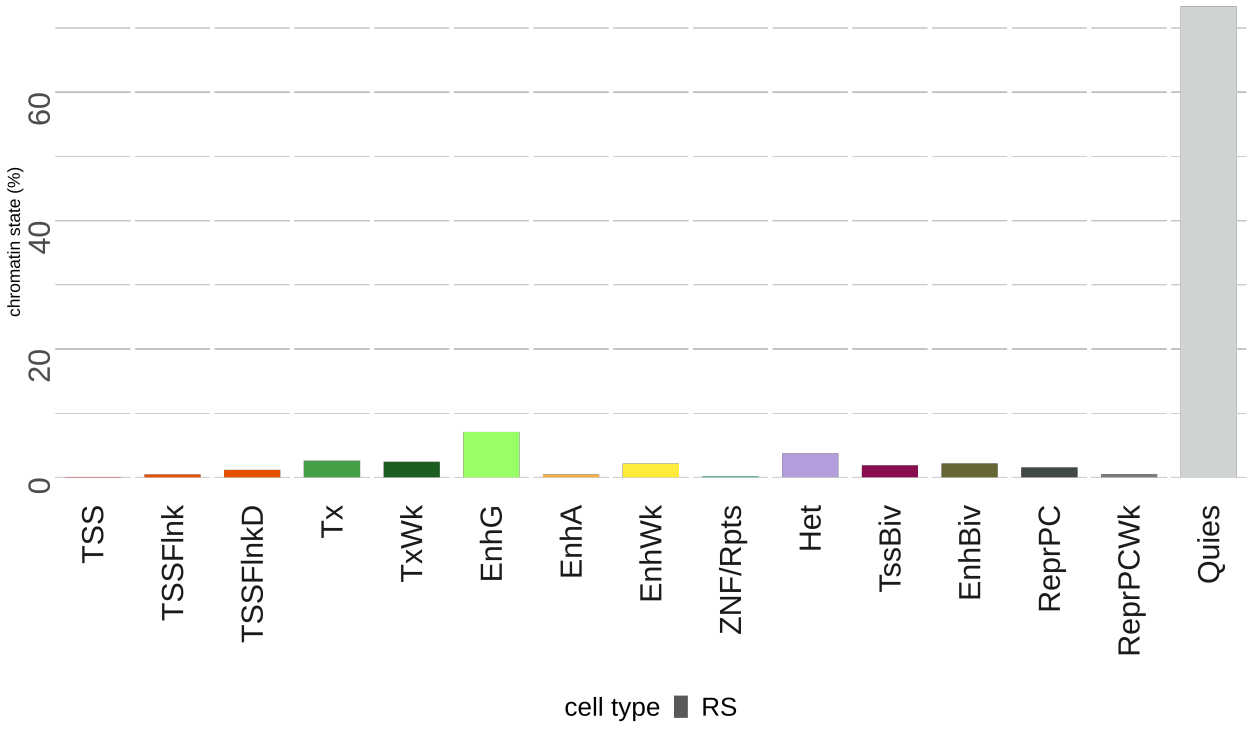
```
#>          <Rle>       <IRanges>  <Rle> | <character> <character>   <factor>
#>   [1]    chr10       0-3100000      * |         U16          RS     Quies
#>   [2]    chr10 3100000-3109200      * |         U11          RS     Het
#>   [3]    chr10 3109200-3110600      * |         U12          RS     TssBiv
#>   [4]    chr10 3110600-3111000      * |         U14          RS     ReprPC
#>   [5]    chr10 3111000-3111200      * |         U13          RS     EnhBiv
#>   [6]    chr10 3111200-3117200      * |         U12          RS     TssBiv
#>   -------
#>   seqinfo: 22 sequences from mm10 genome; no seqlengths
```

Distribution of the chromatin states in the genome

`plotChromatinState()` calculates the percentage of each chromatin state at a given genomic region. The output consists of a dataframe with the percentage of coverage for each chromatin state relatively to the length of the genomic region. It is possible to plot the results in `PNG` file with the argument `plot = TRUE`. If you have a list of dataframe, it is possible to merge all the dataframe in a unique merged dataframe and in a unique plot with the argument `merge = TRUE`.

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

```
#>             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 enhancers

Enhancers are cis-regulatory regions that (locate more or less) near or even within their regulated gene. We assume that an enhancer, may regulate all its neighbouring genes within a given distance (in this present study, the distance is 500kb). We focus on enhancer chromatin states (in this study, we merged them into four types: bivalent enhancers (EnhBiv), genic enhancers (EnhG), active enhancers (EnhA) and weak enhancers (EnhWk)).

`listTableEnhancer` is a GRanges object or a list of GRanges objects (produced by `GenomicRanges` package). Similar to `chromatinState` dataframe, `listTableEnhancer` should contain genes and chromatin states informations. Sample name (sample_name) is mandatory to compare 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: 22 sequences from mm10 genome; no seqlengths
```

Association of enhancers to genes

To determine which genes are associated to which enhancers, we assign to each enhancer all the genes located within an interval. To do that, `enhancerAnnotation()` uses a `GRanges` object.

The function takes few minutes to process depending on the size of your enhancer table. It is possible to preformed multithreading using the `nCore` parameter. To each enhancer position, we obtain the list of associated genes and their distance from the enhancer (in bp).

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

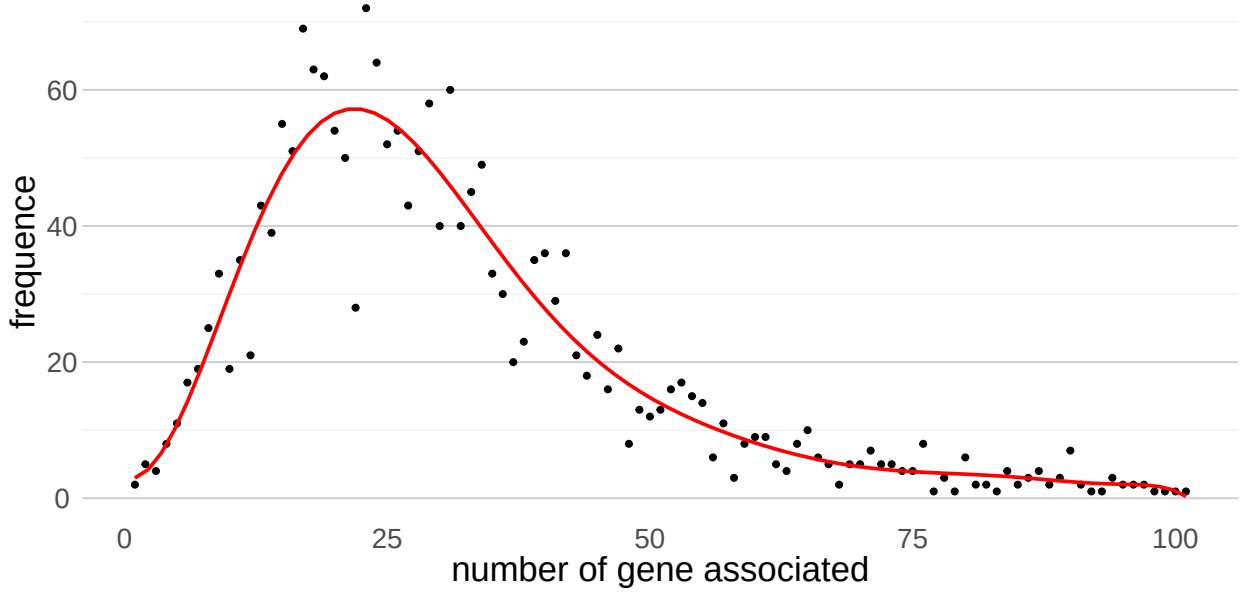
```
#> GRanges object with 6 ranges and 7 metadata columns:
#>     seqnames            ranges strand | chromatin_state sample_name start_500kb
#>        <Rle>         <IRanges>  <Rle> |     <character> <character>   <numeric>
#>   1    chr10   9164400-9164800      * |             U13      EnhBiv     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.. ENSMUSG00000111215.1..
#>   1   9844000               21 456130;480757;457563.. ENSMUSG00000015305.6..
#>   1  10976600               20 499773;435480;392457.. ENSMUSG00000101621.2..
#>   1  21021000               16 371729;318362;311710.. ENSMUSG00000019996.1..
#>   1  21452600               21 227322;432632;326765.. ENSMUSG00000019990.1..
#>   1  21810600               21 430427;356853;275607.. ENSMUSG00000111177.1..
#>   -------
#>   seqinfo: 22 sequences from mm10 genome; no seqlengths
```

Number of genes associated with an enhancer

With the `enhancerAnnotation()` function, each enhancer region can be associated at least one genes. The function `plotGeneAssociation()` allows to represent the distribution of the number of genes associated with the enhancers. The function uses polynomial linear regression for the graph representation.

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



Gene expression information

`geneExpression` is a dataframe that contains information on the gene expression level.

It is generated with the results from RNAseq gene expression analysis. `geneExpression` should contain the following information: chromosome (chr), gene position (start and end), gene name (gene_ENS), strand information (strand), level of gene expression (gene_expression). The score is not necessary for the analysis. For the gene name, the same name than the one used to generate the `genomeFile` dataframe should be used.

```
data(geneExpression)
```

```
#>                gene_ENS   chr     start       end strand score gene_expression
#> 1  ENSMUSG00000000001.4  chr3 108107280 108146146      -     .      27.7106904
#> 2 ENSMUSG00000000028.15 chr16  18780447  18811987      -     .      23.5842993
#> 3 ENSMUSG00000000031.16  chr7 142575529 142578143      -     .       0.9386427
#> 4 ENSMUSG00000000037.16  chrX 161117193 161258213      +     .      14.4548991
#> 5 ENSMUSG00000000049.11 chr11 108343354 108414396      +     .      36.6169129
```

```
#> 6  ENSMUSG00000000056.7 chr11 121237253 121255856      +     .       5.2791187
```

`enhancerExpression()` is able to associate the level of gene expression to each gene-enhancer pair that was determined by the `enhancerAnnotation` function. When a gene-enhancer pair is not associated to an expression level, the function indicates NA.

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

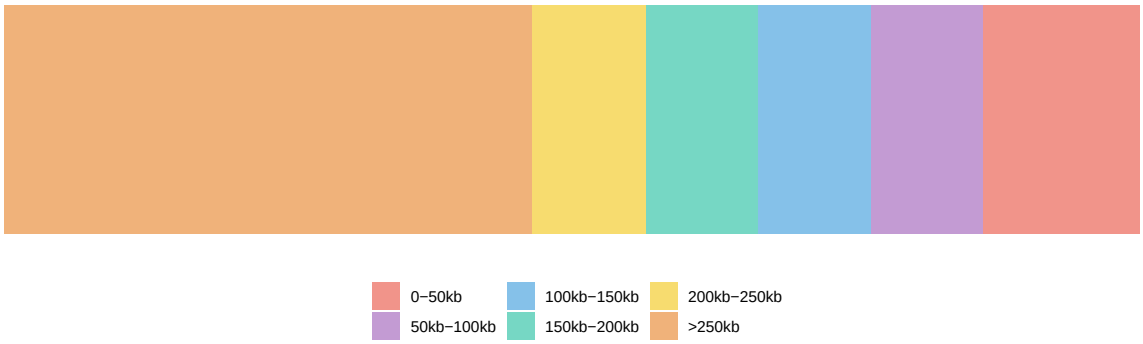
```
#> GRanges object with 6 ranges and 8 metadata columns:
#>     seqnames            ranges strand | chromatin_state sample_name start_500kb
#>        <Rle>         <IRanges>  <Rle> |     <character> <character>   <numeric>
#>   1    chr10   9164400-9164800      * |             U13      EnhBiv     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.. ENSMUSG00000111215.1..
#>   1   9844000               21 456130;480757;457563.. ENSMUSG00000015305.6..
#>   1  10976600               20 499773;435480;392457.. ENSMUSG00000101621.2..
#>   1  21021000               16 371729;318362;311710.. ENSMUSG00000019996.1..
#>   1  21452600               21 227322;432632;326765.. ENSMUSG00000019990.1..
#>   1  21810600               21 430427;356853;275607.. ENSMUSG00000111177.1..
#>            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: 22 sequences from mm10 genome; no seqlengths
```

Visualization of enhancer annotation

Distribution of genes according to their distance from the enhancer

`plotGeneDistance()` enables the generation of a plot showing gene distribution according to their distance from the associated enhancer. The distance is calculated using the `limit` argument and clustered into six groups as shown in the plot below.

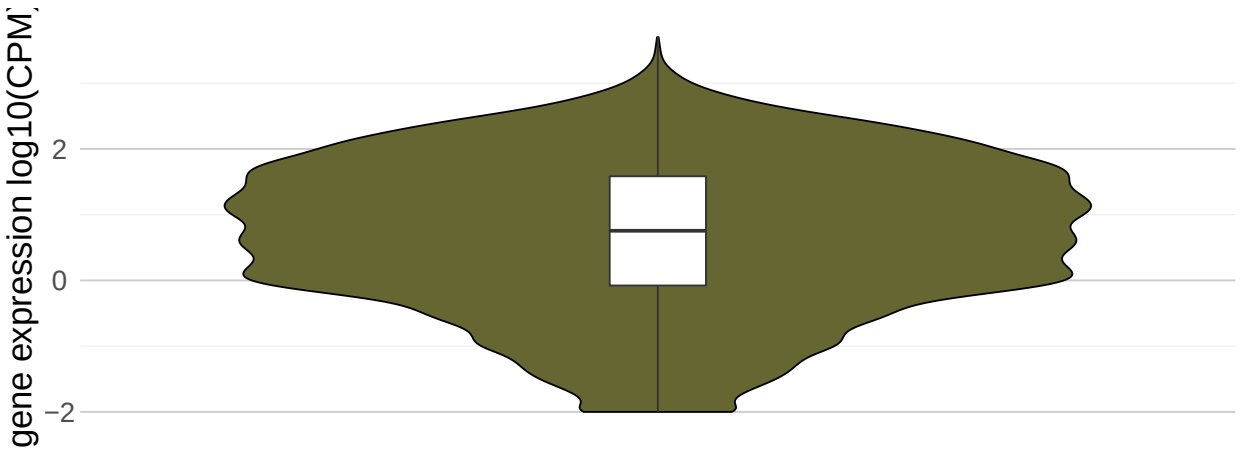
```
plotGeneDistance(table_enhancer_gene_expression, limit = 500000, xlab = "",
ylab = "distance enhancer-gene (bp)")
```



Expression of a gene associated with a given enhancer

`plotEnhancerExpression()` allows to generate a plot of gene expression distribution according to the type of enhancer. It is possible to rescale the plot using the `scale` argument ('none','log10' and 'log2' are accepted).

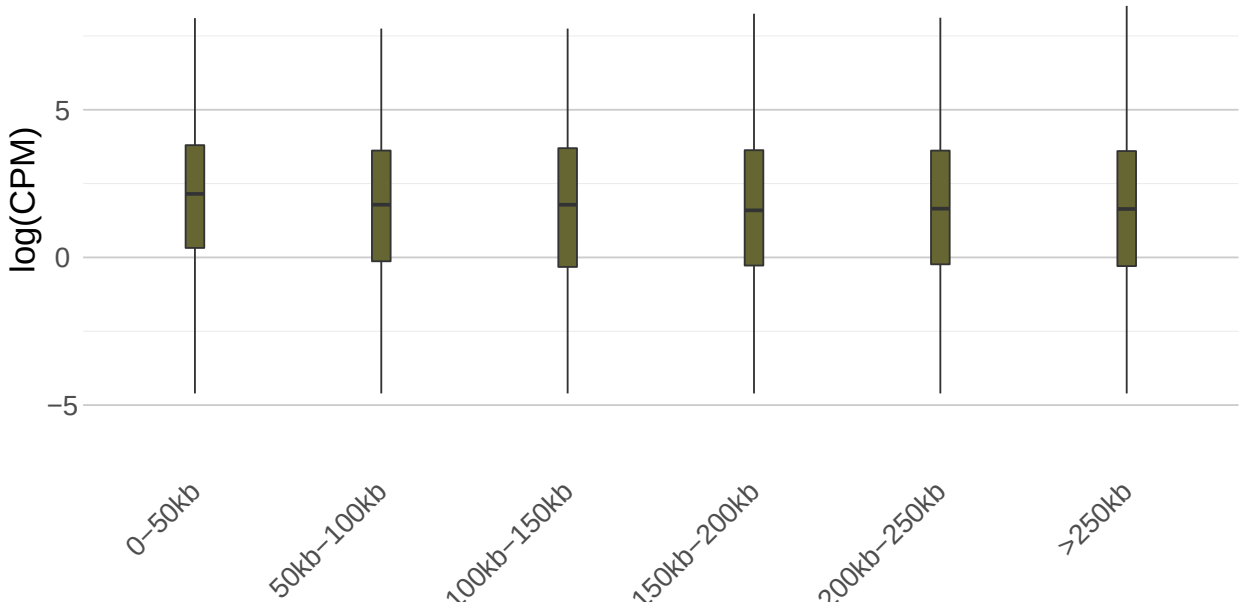
```
plotEnhancerExpression(table_enhancer_gene_expression, scale = "log10",
colorTable = colorTable, ylab = "gene expression log10(CPM)")
```



Gene expression according to gene-enhancer distance

`plotDistanceExpression()` enables the generation of a plot of the level of gene expression according to the gene-enhancer distance. The distance is calculated using `limit` argument and clusterized into six groups as illustrated in the plot below.

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



Enhancer annotation comparison

It is possible to compare different categories of enhancers by means of a list of GRanges objects, each containing input information similar to the one in `listTableEnhancer`. Unlike the individual analysis, each GRanges object in the list requires sample information (sample_name).

The first step is to assign to each enhancer all the genes located within an interval using `enhancerAnnotation()`. After gene association, we associate the gene expression at enhancer using `enhancerExpression()`.

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

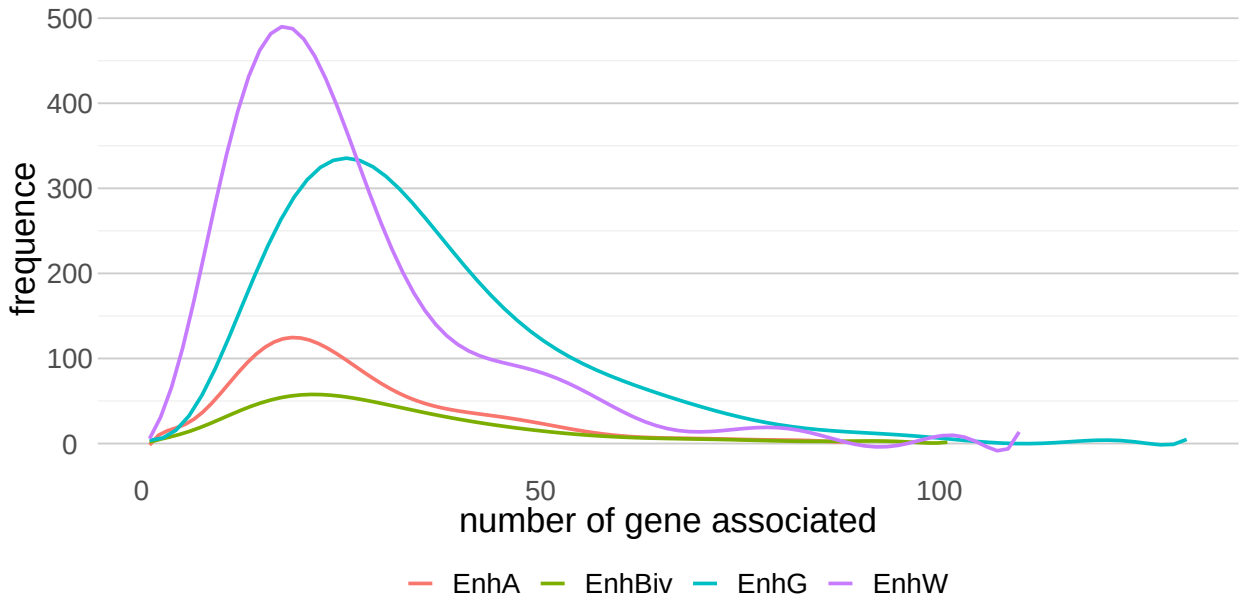
This process takes a few minutes. To reduce time, you can load the `listTableEnhancerGeneExpression` data to process the following analyses.

```
data(listTableEnhancerGeneExpression)
```

Number of genes associated with the enhancer

With the `enhancerAnnotation()` function, each enhancer region can be associated with at least one gene. The function `plotGeneAssociation()` allows to represent the distribution of the number of genes associated with the enhancers. The function uses polynomial linear regression for the graph representation. `all = TRUE` parameter is used to compile all enhancer tables in same '.png' file.

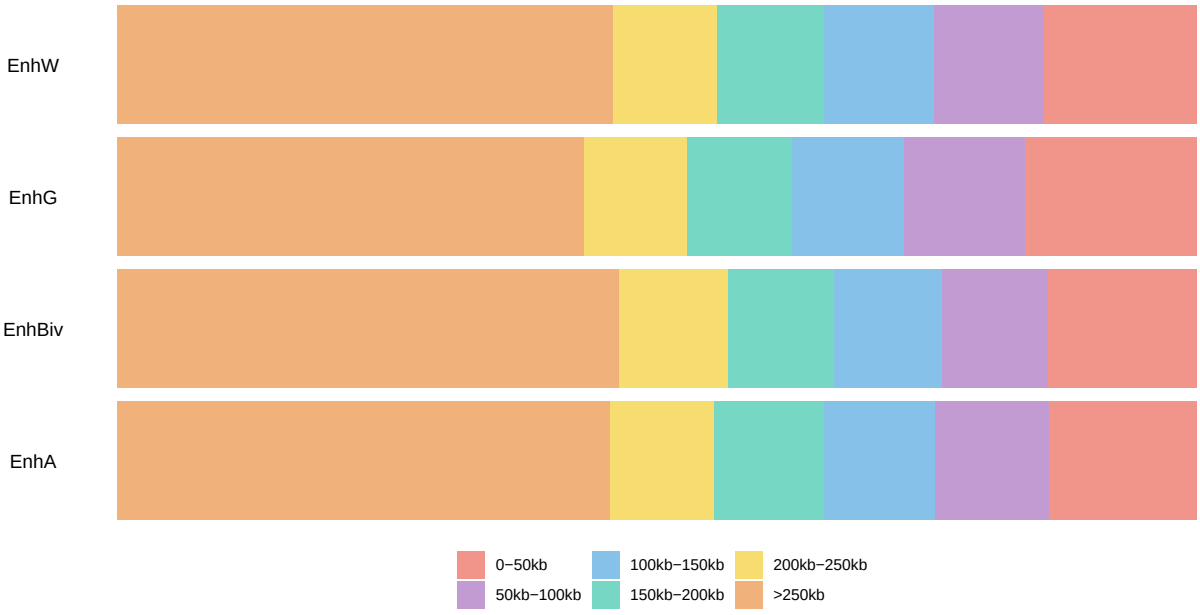
```
plotGeneAssociation(listTableEnhancerGeneExpression, all = TRUE)
```



Distribution of genes according to the gene-enhancer distance

`plotGeneDistance()` allows to generate a plot of gene distribution according to gene-enhancer distance. The distance is calculated with the `limit` argument and clustered into six groups as illustrated in the plot below.

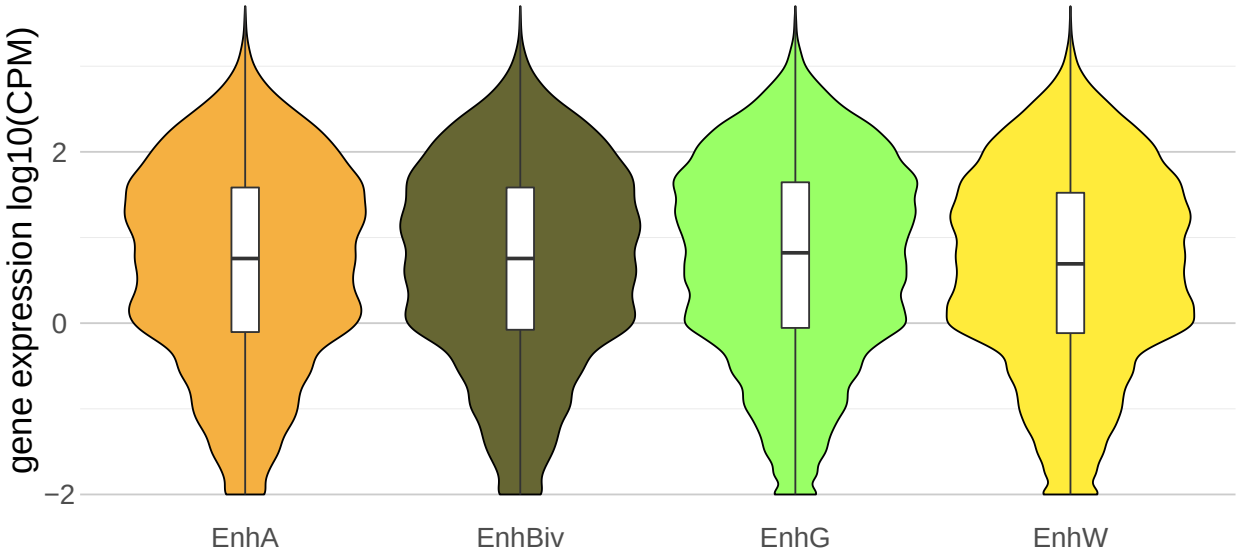
```
plotGeneDistance(listTableEnhancerGeneExpression, limit = 500000,
xlab = "", ylab = "distance enhancer-gene (bp)")
```



Expression of a gene associated with enhancers

`plotEnhancerExpression()` allows to generate a plot of gene expression distribution according to the type of enhancer. It is possible to rescale the plot using the `scale` argument ('none','log10' and 'log2' are accepted).

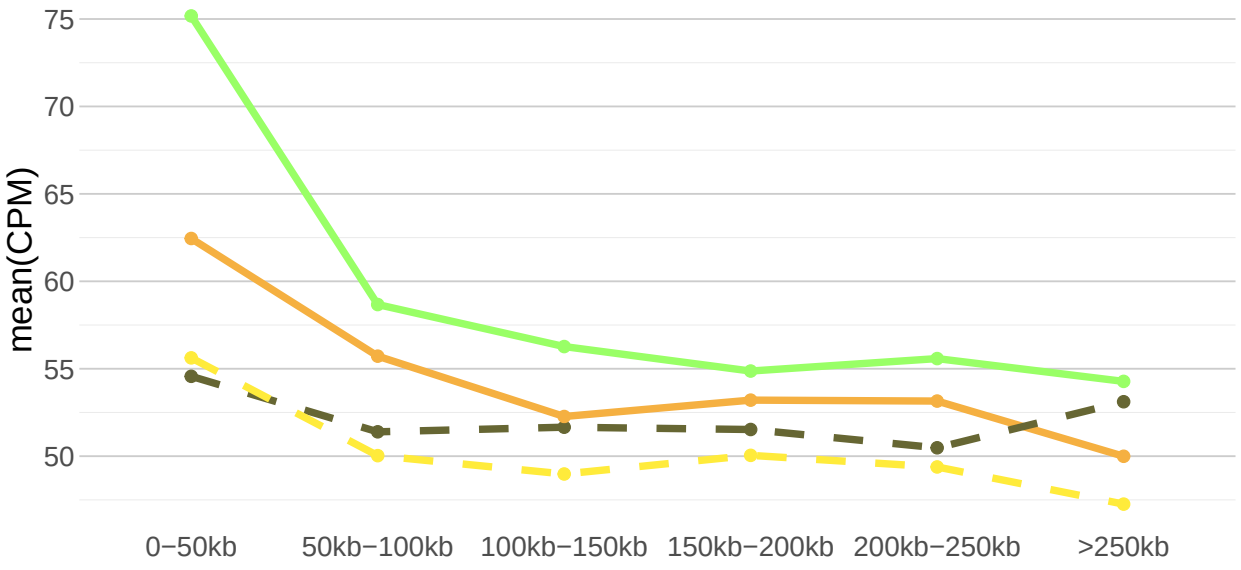
```
plotEnhancerExpression(listTableEnhancerGeneExpression, scale = "log10",
colorTable = colorTable, ylab = "gene expression log10(CPM)")
```



Expression of genes according to their distance from their associated enhancers

This function generates a plot to visualize the level of gene expression according to the distance between a gene and its associated enhancer, using `plotDistanceExpression`. The distance is calculated using the `limit` argument and clustered into six groups as shown in the plot below. In case of list of enhancer, the function shows the average expression of all genes associated with each enhancer.

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



Characterization of chromatin states in the gene environment

This aims at analyzing the chromatin landscape within genes. To perform this analysis, gene expression data from RNAseq analysis (`geneExpression`) as well as chromatin state data from ChromHMM analysis (`chromatinState`) are needed.

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

Chromatin states at gene promoters

The `geneEnvironment()`function calculates the percentage of overlap of each chromatin state with each genes promoters using the `interval` parameter.

`geneEnvironment()` may take a few minutes depending on the number of genes analyzed.

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

```
#>                gene_ENS   chr     start       end strand score gene_expression
#> 1  ENSMUSG00000000001.4  chr3 108107280 108146146      -     .      27.7106904
#> 2 ENSMUSG00000000028.15 chr16  18780447  18811987      -     .      23.5842993
#> 3 ENSMUSG00000000031.16  chr7 142575529 142578143      -     .       0.9386427
#> 4 ENSMUSG00000000037.16  chrX 161117193 161258213      +     .      14.4548991
#> 5 ENSMUSG00000000049.11 chr11 108343354 108414396      +     .      36.6169129
#> 6  ENSMUSG00000000056.7 chr11 121237253 121255856      +     .       5.2791187
#>         TSS TSS_moins_3kb TSS_plus_3kb       TSSA    TSSFlnk TSSFlnkD Tx TxWk
#> 1 108146146     108143146    108149146 0.00000000 0.00000000        0  0    0
#> 2  18811987      18808987     18814987 0.00000000 0.06666667        0  0    0
#> 3 142578143     142575143    142581143 0.00000000 0.00000000        0  0    0
#> 4 161117193     161114193    161120193 0.03333333 0.40000000        0  0    0
#> 5 108343354     108340354    108346354 0.00000000 0.00000000        0  0    0
#> 6 121237253     121234253    121240253 0.00000000 0.06666667        0  0    0
#>        EnhG EnhA  EnhWk ZNF.Rpts Het    TssBiv EnhBiv ReprPC ReprPCWk     Quies
#> 1 0.7423333    0 0.0000        0   0 0.2576667    0.0 0.0000        0 0.0000000
#> 2 0.6333333    0 0.0000        0   0 0.3000000    0.0 0.0000        0 0.0000000
#> 3 0.0000000    0 0.0000        0   0 0.0000000    0.4 0.3095        0 0.2905000
#> 4 0.0000000    0 0.1655        0   0 0.0000000    0.0 0.0000        0 0.4011667
#> 5 0.0000000    0 0.0000        0   0 0.3000000    0.3 0.3410        0 0.0590000
#> 6 0.6000000    0 0.0000        0   0 0.3333333    0.0 0.0000        0 0.0000000
```

Predominant chromatin state at gene promoters

`predominantState()` estimates the predominant chromatin state at gene promoter, which corresponds to the state with the largest overlap with the gene promoter environment. Genes are then clustered according to their chromatin state using `umap` package. The outputcontains information on the predominant chromatin state and the corresponding 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
```

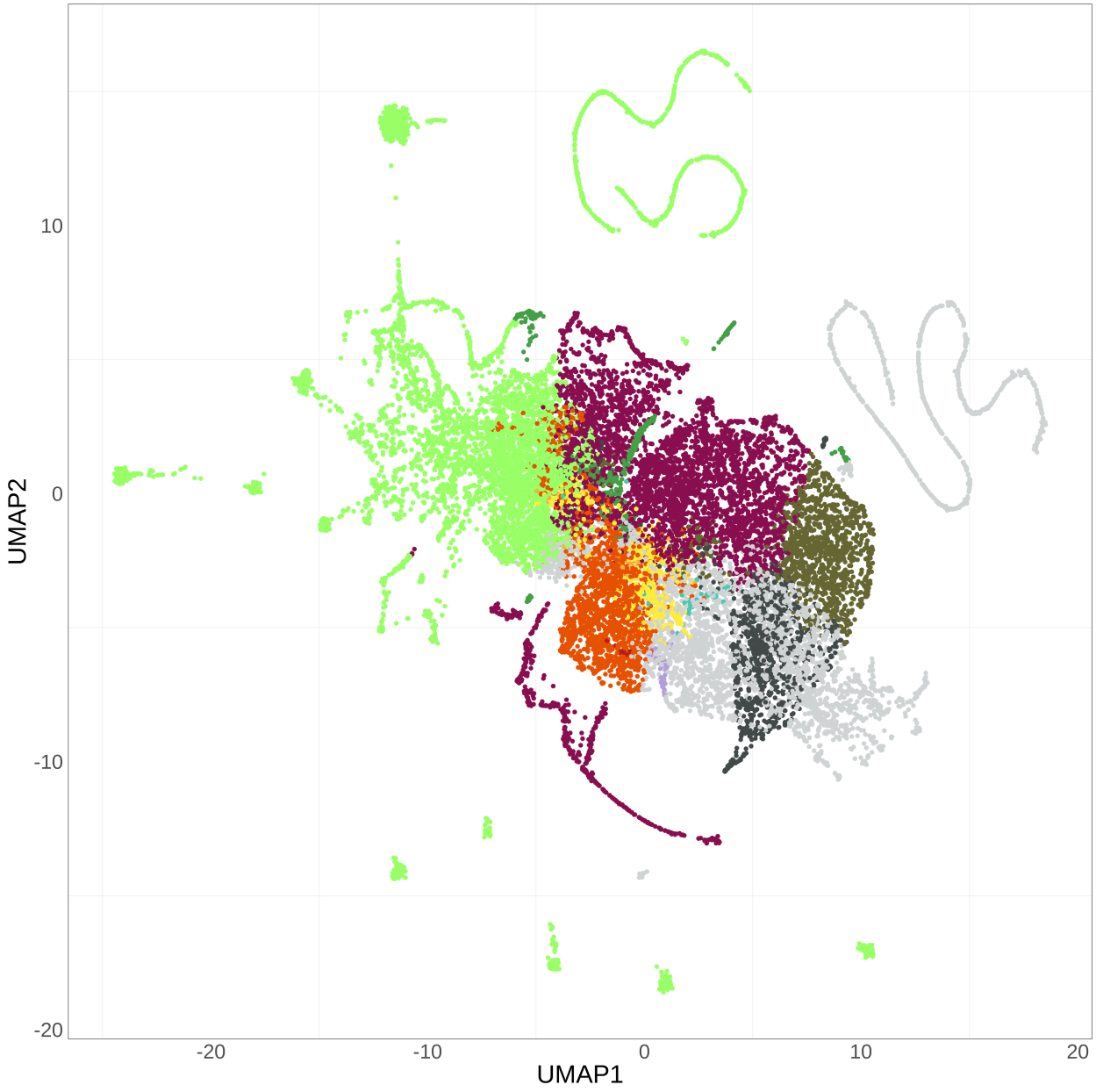
```
#>         TSSA    TSSFlnk TSSFlnkD Tx TxWk      EnhG EnhA  EnhWk ZNF.Rpts Het
#> 1 0.00000000 0.00000000        0  0    0 0.7423333    0 0.0000        0   0
```

```
#> 2 0.00000000 0.06666667        0  0    0 0.6333333    0 0.0000        0   0
#> 3 0.00000000 0.00000000        0  0    0 0.0000000    0 0.0000        0   0
#> 4 0.03333333 0.40000000        0  0    0 0.0000000    0 0.1655        0   0
#> 5 0.00000000 0.00000000        0  0    0 0.0000000    0 0.0000        0   0
#> 6 0.00000000 0.06666667        0  0    0 0.6000000    0 0.0000        0   0
#>      TssBiv EnhBiv ReprPC ReprPCWk     Quies       UMAP1     UMAP2  state
#> 1 0.2576667    0.0 0.0000        0 0.0000000 -8.67742450 13.346519   EnhG
#> 2 0.3000000    0.0 0.0000        0 0.0000000  0.17876276  9.923848   EnhG
#> 3 0.0000000    0.4 0.3095        0 0.2905000 -5.54580147 -6.817452 EnhBiv
#> 4 0.0000000    0.0 0.0000        0 0.4011667  2.00365717 -5.308925  Quies
#> 5 0.3000000    0.3 0.3410        0 0.0590000 -4.74353928 -5.283229 ReprPC
#> 6 0.3333333    0.0 0.0000        0 0.0000000  0.08402664  9.072409   EnhG
```

Below is an example of UMAP representation to visualize the predominant chromatin state in each gene. Each dot corresponds to a gene and is colored according to its predominant chromatin state. The resulting figure may not be exactly the same than the one presented in this thumbnail since the order of display?? of dimension axes may differ; however, the clusters remain the same.

Here is an example of code to generate the figure below:

```
ggplot(result_umap,aes(UMAP1,UMAP2, color = factor(state,
    levels = unique(colorTable$stateName)))) +
    geom_point() +
    scale_color_manual(values = colorTable$colorValue) +
    theme_bw() + theme(strip.background  = element_blank(),
        text = element_text(size=25, angle = 0),
        panel.grid.major = element_blank(),
        axis.ticks = element_blank(),
        strip.text.x = element_text(size = 25, angle = 0, hjust = 1),
        legend.position = "none")
```



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/libopenblasp-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] stats     graphics  grDevices utils     datasets  methods   base
#>
#> other attached packages:
#> [1] ChromENVEE_0.99.8
#>
#> 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           grid_4.1.3             nlme_3.1-158
#> [52] jsonlite_1.8.0         gtable_0.3.1           lifecycle_1.0.2
#> [55] DBI_1.1.3              magrittr_2.0.3         scales_1.2.1
#> [58] cli_3.4.0              stringi_1.7.8          cachem_1.0.6
#> [61] farver_2.1.1           XVector_0.34.0         fs_1.5.2
#> [64] remotes_2.4.2          ellipsis_0.3.2         vctrs_0.4.1
#> [67] generics_0.1.3         tools_4.1.3            glue_1.6.2
#> [70] purrr_0.3.4            processx_3.7.0         pkgload_1.3.0
#> [73] parallel_4.1.3         fastmap_1.1.0          yaml_2.3.5
#> [76] colorspace_2.0-3       GenomicRanges_1.46.1   sessioninfo_1.2.2
#> [79] memoise_2.0.1          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

Ferrari, F. et al. DOT1L-mediated murine neuronal differentiation associates with H3K79me2 accumulation and preserves SOX2-enhancer accessibility. Nat. Commun. 11, 5200 (2020)

Godfrey, L. et al. DOT1L inhibition reveals a distinct subset of enhancers dependent on H3K79 methylation. Nat. Commun. 10, 1–15 (2019).

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/jour-

nal.pcbi.1003118, http://www.ploscompbiol.org/article/info%3Adoi%2F10.1371%2Fjournal.pcbi.1003118.

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