Representing Dimer Transcription Factor Complexes using Forked-Position Weight Matrices

2019-10-18

Abstract

Transcription factors (TFs) Transcription factors (TFs) are key cellular components that control gene expression. They recognize specific DNA sequences, the TF binding sites (TFBSs), and thus are targeted to specific regions of the genome where they can recruit transcriptional co-factors and/or chromatin regulators tone-tune spatiotemporal gene regulation. Therefore, the identification of TFBSs in genomic sequences and their subsequent quantitative modeling is of crucial importance for understanding and predicting gene expression. Most transcription factors do not work alone. Many large TF families form complex homotypic or heterotypic interactions through dimerization. For gene transcription to occur, a number of transcription factors must bind to DNA regulatory sequences. Because of this contribution of multiple TFs, a single sequence matrix/logo is not sufficiently expressing as it is not properly designed to illustrate the impact of other TFs engaged in transcription process. To tackle this issue, we purpose a model which consists a graph of multiple PFMs (or Seq-logos) to have a better representation of a binding site of a TF of interest, in the presence of other factors.

0.1 Matrix Deconvolution

To retrieve initial matrices, we employ TFregulomeR library which is a comprehensive Transcription Factor Binding Site (TFBS) database combining MethMotif and GTRD. TFregulomeR allows users to easily browse the TFregulome database using only one simple function TFBSBrowser. Users can search the database according to species, organ, sample type, cell/tissue name, TF name, disease state, and source.

```
library(TFregulomeR)
k562_numbers <- TFBSBrowser(cell_tissue_name = "k562")
#> 131 TFBS(s) found: ...
#> ... covering 131 TF(s)
#> ... from 1 species:
#> ... ...human
#> ... from 1 organ(s):
#> ... ... blood_and_lymph
#> ... in 1 sample type(s):
#> ... ... cell_line
#> ... in  1  different cell(s) or tissue(s)
#> ... in 1 type(s) of disease state(s):
#> ... ... tumor
#> ... from the source(s): MethMotif
```

TFregulomeR allows users to portray the co-binding landscapes between two collections of TFs, along with DNA methylation states in the pair-wise intersected peaks, using intersectPeakMatrix. This functionality is particularly useful to study TF interactome in a cell type. It perform an exhaustive intersection analysis between a pair of peak sets, one from list x and the other from list y, to form an x*y intersection matrix. Therefore, it is required for users to provide the two lists of peak sets.

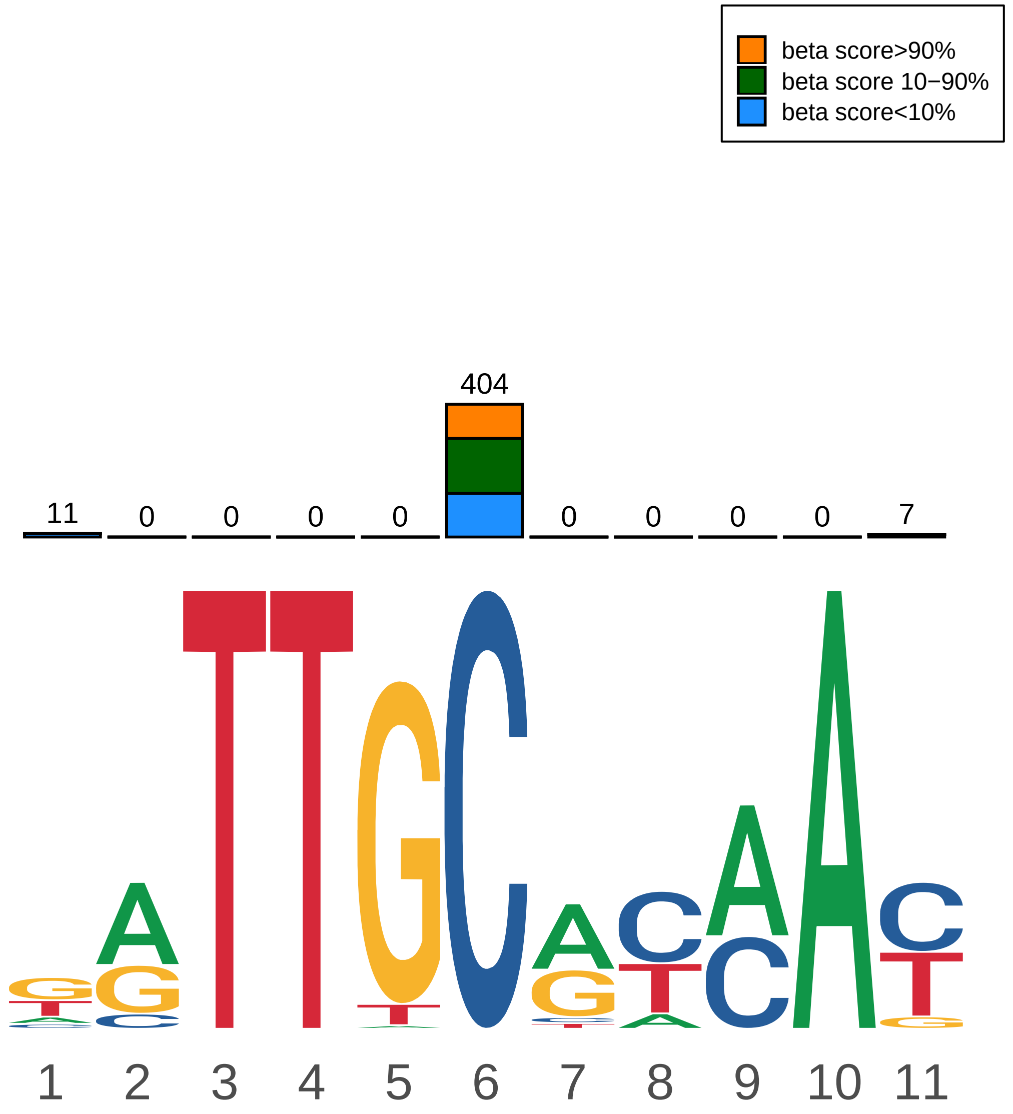
For peak list x, users can directly use TFregulome peaks by providing TFregulome ID in peak_id_x and indicating whether loading peaks with motif only using motif_only_for_id_x. In addition, customized peak sets can also be input in user_peak_list_x. It's recommended that unique IDs (also unique to IDs in peak_id_x) be provided for each customized peak set in user_peak_x_id. If the customized peak set is a TFregulome TF subset, it's highly recommended that the corresponding TFregulome ID be provided in user_peak_x_id, which allows the function to recognizethe source of peak set and to properly profile the DNA methylation states in the intersected regions, if methylation_profile_in_narrow_region=T. Even though

TFregulome peak sets are peak summits, the function is able to recognizeit with the provided TFregulome ID in peak_id_x and automatically expand +/- 100bp during the analysis. Same principles are applicable for peak list y.

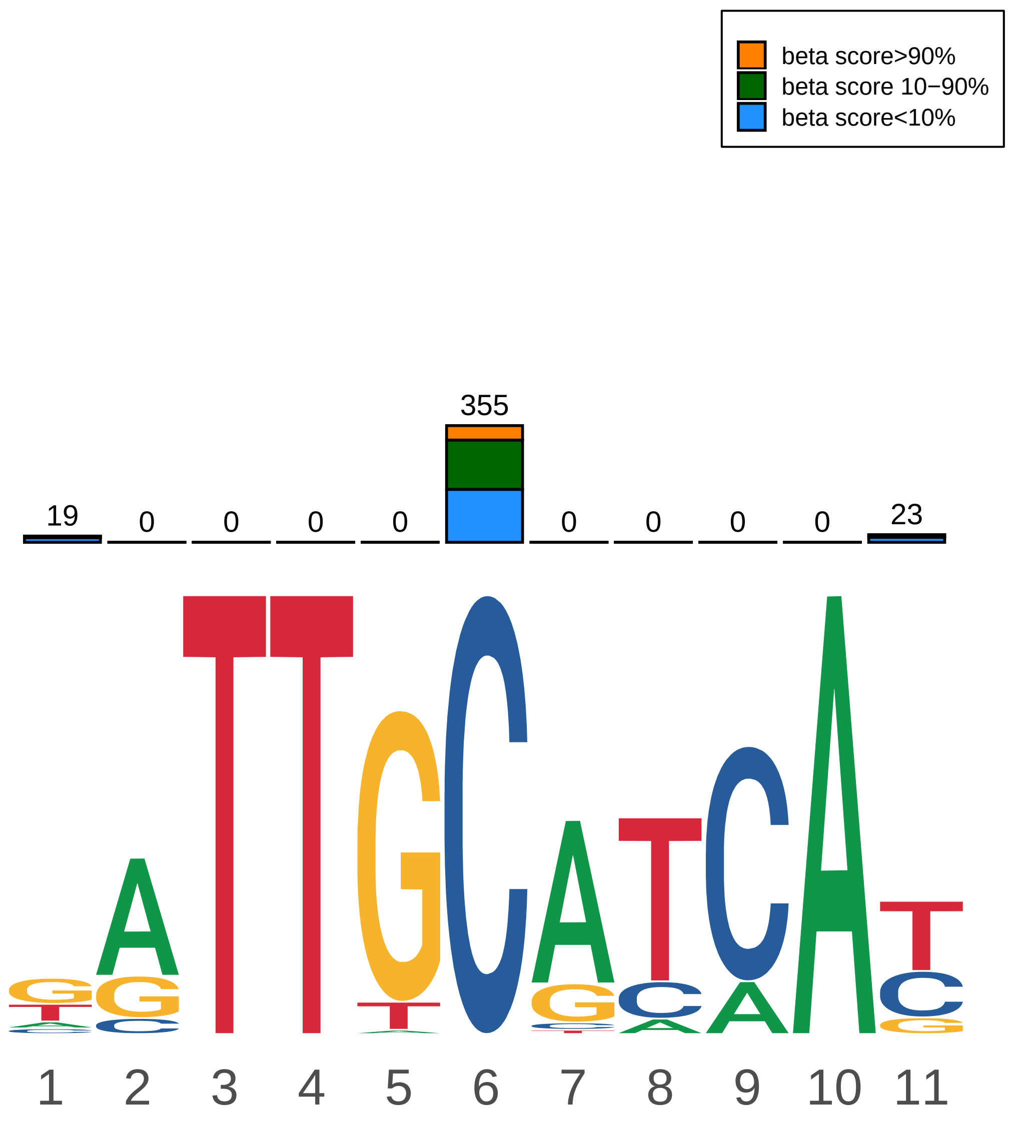
```
CEBPB_CEBPD <- intersectPeakMatrix(peak_id_x = "MM1_HSA_K562_CEBPB", motif_only_for_id_x = T,
    peak_id_y = "MM1_HSA_K562_CEBPD", motif_type = "TRANSFAC")
#> TFregulomeR::intersectPeakMatrix() starting ... ...
#> You chose NOT to profile the methylation levels in 200bp window around peak summits
#> Loading peak list x ... ...
#> ... You have 1 TFBS(s) requested to be loaded from TFregulomeR server
#> ... You chose to load TF peaks with motif only. Using 'motif_only_for_id_x' tunes your options
#> ... loading TFBS(s) from TFregulomeR now
#> ... ... peak file loaded successfully for id 'MM1_HSA_K562_CEBPB'
#> ... Done loading TFBS(s) from TFregulomeR
#> Loading peak list y ... ...
#> ... You have 1 TFBS(s) requested to be loaded from TFregulomeR server
#> ... You chose to load TF peaks regardless of presence of motif. Using 'motif_only_for_id_y' tunes you
#> ... loading TFBS(s) from TFregulomeR now
#> ... ... peak file loaded successfully for id 'MM1_HSA_K562_CEBPD'
#> ... Done loading TFBS(s) from TFregulomeR
#> Start analysing list x:MM1_HSA_K562_CEBPB... ...
#> ... ... Start analysing list y:MM1_HSA_K562_CEBPD
CEBPB_ATF4 <- intersectPeakMatrix(peak_id_x = "MM1_HSA_K562_CEBPB", motif_only_for_id_x = T,
    peak_id_y = "MM1_HSA_K562_ATF4", motif_type = "TRANSFAC")
#> TFregulomeR::intersectPeakMatrix() starting ... ...
#> You chose NOT to profile the methylation levels in 200bp window around peak summits
#> Loading peak list x ... ...
#> ... You have 1 TFBS(s) requested to be loaded from TFregulomeR server
#> ... You chose to load TF peaks with motif only. Using 'motif_only_for_id_x' tunes your options
#> ... loading TFBS(s) from TFregulomeR now
#> ... ... peak file loaded successfully for id 'MM1_HSA_K562_CEBPB'
#> ... Done loading TFBS(s) from TFregulomeR
#> Loading peak list y ... ...
#> ... You have 1 TFBS(s) requested to be loaded from TFregulomeR server
#> ... You chose to load TF peaks regardless of presence of motif. Using 'motif_only_for_id_y' tunes you
#> ... loading TFBS(s) from TFregulomeR now
#> ... ... peak file loaded successfully for id 'MM1_HSA_K562_ATF4'
#> ... Done loading TFBS(s) from TFregulomeR
#> Start analysing list x:MM1_HSA_K562_CEBPB... ...
#> ... ... Start analysing list y:MM1_HSA_K562_ATF4
```

A function named intersectPeakMatrixResult is implemented in TFregulomeR package, allowing an easy extraction and interpretation of intersectPeakMatrix output. It is worth noting that there are two ways of interpretations in intersection between set A and B, that is, 1) the percentage of A overlapped with B, and 2) the percentage of B intersected with A. The option is usually up to which study object should be focused on. Same principle is applicable for the output of intersectPeakMatrix.

```
results_CEBPB_CEBPD <- intersectPeakMatrixResult(intersectPeakMatrix = CEBPB_CEBPD,
    return_intersection_matrix = T, save_MethMotif_logo = T)
#> Start getting the results of intersectPeakMatrix ...
#> ... ... You chose to return intersection matrix;
#> ... ... ... You chose x-wise intersection matrix;
#> ... ... You chose NOT to return tag density;
#> ... ... You chose NOT to return methylation profile;
#> ... ... You chose to save MethMotif logo in PDF if any;
#> ... ... ... You chose x-wise MethMotif logo;
#> ... ... ... You chose entropy logo;
#> ... ... ... You chose to show all methylation levels;
#> Success: a PDF named 'MM1_HSA_K562_CEBPB_overlapped_with_MM1_HSA_K562_CEBPD-logo-entropy.pdf' has bee
```



```
results_CEBPB_ATF4 <- intersectPeakMatrixResult(intersectPeakMatrix = CEBPB_ATF4,
    return_intersection_matrix = T, save_MethMotif_logo = T)
#> Start getting the results of intersectPeakMatrix ...
#> ... ... You chose to return intersection matrix;
#> ... ... ... You chose x-wise intersection matrix;
#> ... ... You chose NOT to return tag density;
#> ... ... You chose NOT to return methylation profile;
#> ... ... You chose to save MethMotif logo in PDF if any;
#> ... ... ... You chose x-wise MethMotif logo;
#> ... ... ... You chose entropy logo;
#> ... ... ... You chose to show all methylation levels;
#> Success: a PDF named 'MM1_HSA_K562_CEBPB_overlapped_with_MM1_HSA_K562_ATF4-logo-entropy.pdf' has been
```



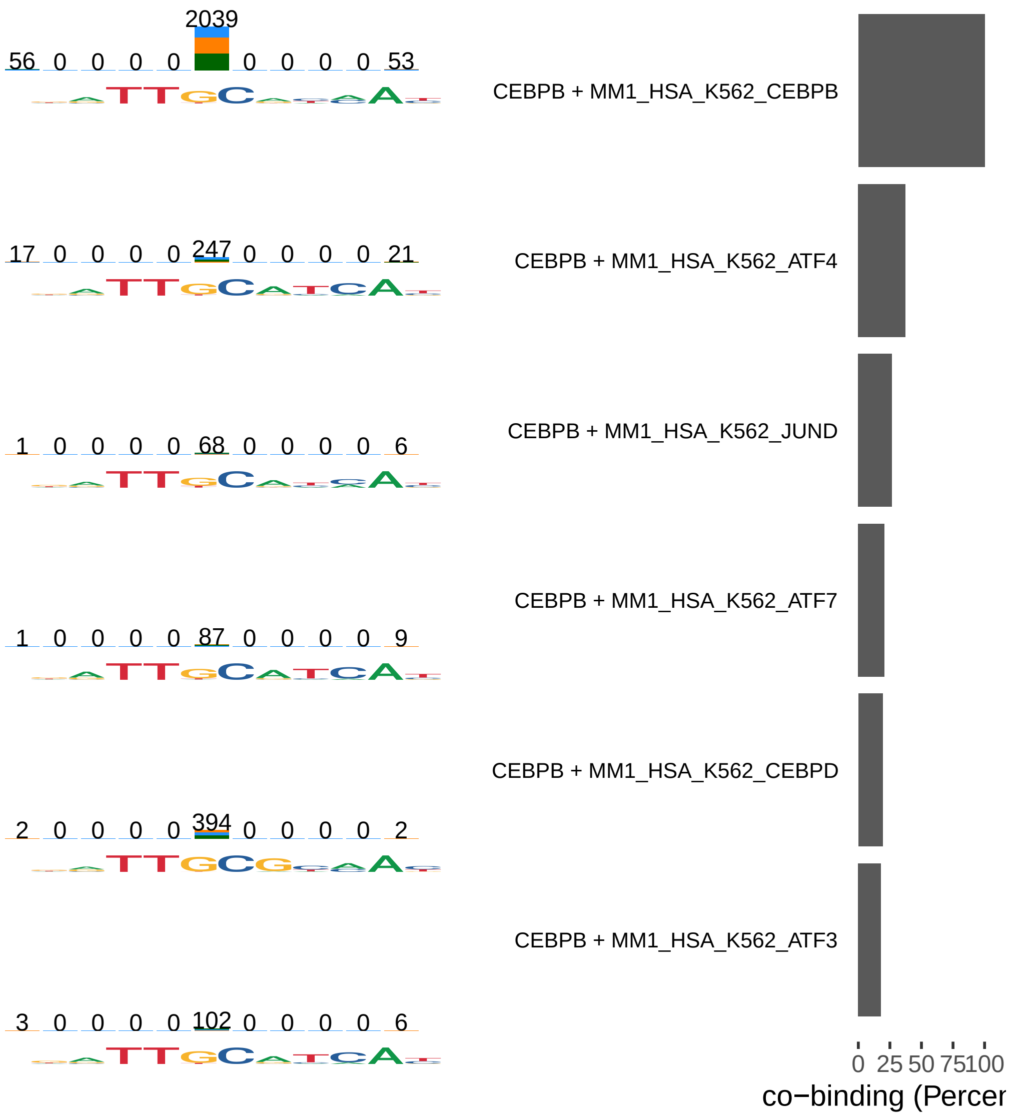
So by setting the value of return_intersection_matrix, save_MethMotif_logo and save_betaScore_matrix as TRUE, we choose to retrieve the logo and matrix of selected data. These two can be saved locally using exportMMPFM.

```
r eval = TRUE
}
exportMMPFM(
fun_output = CEBPB_CEBPD,
fun = "intersectPeakMatrix",
save_motif_PFM = T,
save_betaScore_matrix = T
)
exportMMPFM(
fun_output = CEBPB_ATF4,
fun = "intersectPeakMatrix",
save_motif_PFM = T,
save_betaScore_matrix = T
)
```

0.2 Intersection percentages and Sequence Logos

The sequence logos depicted in FPWM are the result of intersectionPeakMatrix. Each matrix has an overlapping score between two sets of TF peaks. This score is of high importance when it comes to storing TF factors by their significance in our study. FPWM delivers a function that plots Sequence Logos respecting the overlapping scores. User has the option of specifying number of intersected matrices to be shown and setting a limit for presented Overlapping scores. The function takes 6 arguments. Users can input cell type and target Transcription Factor's name so that the function could import information about target TF and all the other TFs present in that cell type. Note that analyzing time for this procedure can take a while regarding the number of TFs and the analyzing time. So, the function can be optionally set to read a local file by setting the argument "Local" to TRUE and providing the path of .csv file to path argument. Also, the function will store a local file named "cp-factors.csv" if it Local==FALSE for future references. Also, the argument "Numberoftop" takes the number of top N logos the user wishes to observe. "Highestscore" on the other hand, sets a limitation on scores in such an order that only logos with a score equal or higher than that will appear. Note than function prioritizes "Numberoftop" over "Highestscore".

```
library(stringr)
FPWM::Barandseqlogo(NumberofTop = 6, highestscore = 10, cell = "k562", TF = "CEBPB",
    Local = TRUE, path = "co-factors.csv", Methylation = TRUE)
#>
#>
#> You chose to work with locally available .CSV file
#> Number of co-factors with co-binding Percentage higher than provided, is more than or equal to selec
#> ...... Showing 6 top ones...
#> TFregulomeR::intersectPeakMatrix() starting ... ...
#> You chose NOT to profile the methylation levels in 200bp window around peak summits
#> Loading peak list x ... ...
#> ... You have 1 TFBS(s) requested to be loaded from TFregulomeR server
#> ... You chose to load TF peaks with motif only. Using 'motif_only_for_id_x' tunes your options
#> ... loading TFBS(s) from TFregulomeR now
#> ... ... peak file loaded successfully for id 'MM1_HSA_k562_CEBPB'
#> ... Done loading TFBS(s) from TFregulomeR
#> Loading peak list y ... ...
#> ... You have 1 TFBS(s) requested to be loaded from TFregulomeR server
#> ... You chose to load TF peaks with motif only. Using 'motif_only_for_id_y' tunes your options
#> ... loading TFBS(s) from TFregulomeR now
#> ... ... peak file loaded successfully for id 'MM1_HSA_K562_CEBPB'
#> ... Done loading TFBS(s) from TFregulomeR
#> Start analysing list x:MM1_HSA_k562_CEBPB... ...
#> ... ... Start analysing list y:MM1_HSA_K562_CEBPB
#> TFregulomeR::intersectPeakMatrix() starting ... ...
#> You chose NOT to profile the methylation levels in 200bp window around peak summits
#> Loading peak list x ... ...
#> ... You have 1 TFBS(s) requested to be loaded from TFregulomeR server
#> ... You chose to load TF peaks with motif only. Using 'motif_only_for_id_x' tunes your options
#> ... loading TFBS(s) from TFregulomeR now
#> ... ... peak file loaded successfully for id 'MM1_HSA_k562_CEBPB'
#> ... Done loading TFBS(s) from TFregulomeR
#> Loading peak list y ... ...
#> ... You have 1 TFBS(s) requested to be loaded from TFregulomeR server
#> ... You chose to load TF peaks with motif only. Using 'motif_only_for_id_y' tunes your options
#> ... loading TFBS(s) from TFregulomeR now
#> ... ... peak file loaded successfully for id 'MM1_HSA_K562_ATF4'
#> ... Done loading TFBS(s) from TFregulomeR
#> Start analysing list x:MM1_HSA_k562_CEBPB... ...
#> ... ... Start analysing list y:MM1_HSA_K562_ATF4
#> TFregulomeR::intersectPeakMatrix() starting ... ...
#> You chose NOT to profile the methylation levels in 200bp window around peak summits
#> Loading peak list x ... ...
#> ... You have 1 TFBS(s) requested to be loaded from TFregulomeR server
#> ... You chose to load TF peaks with motif only. Using 'motif_only_for_id_x' tunes your options
#> ... loading TFBS(s) from TFregulomeR now
#> ... ... peak file loaded successfully for id 'MM1_HSA_k562_CEBPB'
#> ... Done loading TFBS(s) from TFregulomeR
#> Loading peak list y ... ...
#> ... You have 1 TFBS(s) requested to be loaded from TFregulomeR server
#> ... You chose to load TF peaks with motif only. Using 'motif_only_for_id_y' tunes your options
#> ... loading TFBS(s) from TFregulomeR now
#> ... ... peak file loaded successfully for id 'MM1_HSA_K562_JUND'
#> ... Done loading TFBS(s) from TFregulomeR
#> Start analysing list x:MM1_HSA_k562_CEBPB... ...
#> ... ... Start analysing list y:MM1_HSA_K562_JUND
#> TFregulomeR::intersectPeakMatrix() starting ... ...
#> You chose NOT to profile the methylation levels in 200bp window around peak summits
#> Loading peak list x ... ...
#> ... You have 1 TFBS(s) requested to be loaded from TFregulomeR server
#> ... You chose to load TF peaks with motif only. Using 'motif_only_for_id_x' tunes your options
#> ... loading TFBS(s) from TFregulomeR now
#> ... ... peak file loaded successfully for id 'MM1_HSA_k562_CEBPB'
#> ... Done loading TFBS(s) from TFregulomeR
#> Loading peak list y ... ...
#> ... You have 1 TFBS(s) requested to be loaded from TFregulomeR server
#> ... You chose to load TF peaks with motif only. Using 'motif_only_for_id_y' tunes your options
#> ... loading TFBS(s) from TFregulomeR now
#> ... ... peak file loaded successfully for id 'MM1_HSA_K562_ATF7'
#> ... Done loading TFBS(s) from TFregulomeR
#> Start analysing list x:MM1_HSA_k562_CEBPB... ...
#> ... ... Start analysing list y:MM1_HSA_K562_ATF7
#> TFregulomeR::intersectPeakMatrix() starting ... ...
#> You chose NOT to profile the methylation levels in 200bp window around peak summits
#> Loading peak list x ... ...
#> ... You have 1 TFBS(s) requested to be loaded from TFregulomeR server
#> ... You chose to load TF peaks with motif only. Using 'motif_only_for_id_x' tunes your options
#> ... loading TFBS(s) from TFregulomeR now
#> ... ... peak file loaded successfully for id 'MM1_HSA_k562_CEBPB'
#> ... Done loading TFBS(s) from TFregulomeR
#> Loading peak list y ... ...
#> ... You have 1 TFBS(s) requested to be loaded from TFregulomeR server
#> ... You chose to load TF peaks with motif only. Using 'motif_only_for_id_y' tunes your options
#> ... loading TFBS(s) from TFregulomeR now
#> ... ... peak file loaded successfully for id 'MM1_HSA_K562_CEBPD'
#> ... Done loading TFBS(s) from TFregulomeR
#> Start analysing list x:MM1_HSA_k562_CEBPB... ...
#> ... ... Start analysing list y:MM1_HSA_K562_CEBPD
#> TFregulomeR::intersectPeakMatrix() starting ... ...
#> You chose NOT to profile the methylation levels in 200bp window around peak summits
#> Loading peak list x ... ...
#> ... You have 1 TFBS(s) requested to be loaded from TFregulomeR server
#> ... You chose to load TF peaks with motif only. Using 'motif_only_for_id_x' tunes your options
#> ... loading TFBS(s) from TFregulomeR now
#> ... ... peak file loaded successfully for id 'MM1_HSA_k562_CEBPB'
#> ... Done loading TFBS(s) from TFregulomeR
#> Loading peak list y ... ...
#> ... You have 1 TFBS(s) requested to be loaded from TFregulomeR server
#> ... You chose to load TF peaks with motif only. Using 'motif_only_for_id_y' tunes your options
#> ... loading TFBS(s) from TFregulomeR now
#> ... ... peak file loaded successfully for id 'MM1_HSA_K562_ATF3'
#> ... Done loading TFBS(s) from TFregulomeR
#> Start analysing list x:MM1_HSA_k562_CEBPB... ...
#> ... ... Start analysing list y:MM1_HSA_K562_ATF3
```



Generating class object of the two target matrices As it was shown in previous section, user can generate an intersecPeakMatrix using two TFregulomeR IDs. In order to study the behavior of TF dimers, it is enlightening to investigate a TF factor's sequence preference in different dimerization. ClassAssignment() function build under "FPWM" package, enables user to generate an object of class 4, which segregates two matrices of two different partners of target TF into three sub matrices. One, for common region, and two others for

exclusive impact of each partner. This segregation is also applied in same method on the associated beta Score matrices. For this purpose, user needs to define the desired function by assigning a character to the argument "fun". Then, a pair of TFregulomeR IDs are needed for the generation of each matrix. Note that one of IDs is common between two matrices, which is the ID of TF of interest. This common ID is defined by argument "peak_id_x". Then two different partners' IDs are assigned to peak_id_y1 and peak_id_y2, similar to the way it was shown how to employ intersectPeakMatrix() function. Finally, a forking point is given to function, from which the final matrix is going to be splitted into two exclusive sub-matrices.

```
library(FPWM)
AclassObject <- FPWM::ObjectGenerator(peak_id_x = "MM1_HSA_K562_CEBPB", peak_id_y_list = list("MM1_HSA_
    "MM1_HSA_K562_ATF4", "MM1_HSA_K562_ATF7"), sp = 5)
#>
#> ....Accessing TFregulomeR!....
#> TFregulomeR::intersectPeakMatrix() starting ... ...
#> You chose NOT to profile the methylation levels in 200bp window around peak summits
#> Loading peak list x ... ...
#> ... You have 1 TFBS(s) requested to be loaded from TFregulomeR server
#> ... You chose to load TF peaks with motif only. Using 'motif_only_for_id_x' tunes your options
#> ... loading TFBS(s) from TFregulomeR now
#> ... ... peak file loaded successfully for id 'MM1_HSA_K562_CEBPB'
#> ... Done loading TFBS(s) from TFregulomeR
#> Loading peak list y ... ...
#> ... You have 1 TFBS(s) requested to be loaded from TFregulomeR server
#> ... You chose to load TF peaks regardless of presence of motif. Using 'motif_only_for_id_y' tunes you
#> ... loading TFBS(s) from TFregulomeR now
#> ... ... peak file loaded successfully for id 'MM1_HSA_K562_CEBPD'
#> ... Done loading TFBS(s) from TFregulomeR
#> Start analysing list x:MM1_HSA_K562_CEBPB... ...
#> ... ... Start analysing list y:MM1_HSA_K562_CEBPD
#>
#> ....Accessing TFregulomeR!....
#> TFregulomeR::intersectPeakMatrix() starting ... ...
#> You chose NOT to profile the methylation levels in 200bp window around peak summits
#> Loading peak list x ... ...
#> ... You have 1 TFBS(s) requested to be loaded from TFregulomeR server
#> ... You chose to load TF peaks with motif only. Using 'motif_only_for_id_x' tunes your options
#> ... loading TFBS(s) from TFregulomeR now
#> ... ... peak file loaded successfully for id 'MM1_HSA_K562_CEBPB'
#> ... Done loading TFBS(s) from TFregulomeR
#> Loading peak list y ... ...
#> ... You have 1 TFBS(s) requested to be loaded from TFregulomeR server
#> ... You chose to load TF peaks regardless of presence of motif. Using 'motif_only_for_id_y' tunes you
#> ... loading TFBS(s) from TFregulomeR now
#> ... ... peak file loaded successfully for id 'MM1_HSA_K562_ATF4'
#> ... Done loading TFBS(s) from TFregulomeR
#> Start analysing list x:MM1_HSA_K562_CEBPB... ...
#> ... ... Start analysing list y:MM1_HSA_K562_ATF4
#>
#> ....Accessing TFregulomeR!....
#> TFregulomeR::intersectPeakMatrix() starting ... ...
#> You chose NOT to profile the methylation levels in 200bp window around peak summits
#> Loading peak list x ... ...
#> ... You have 1 TFBS(s) requested to be loaded from TFregulomeR server
#> ... You chose to load TF peaks with motif only. Using 'motif_only_for_id_x' tunes your options
#> ... loading TFBS(s) from TFregulomeR now
#> ... ... peak file loaded successfully for id 'MM1_HSA_K562_CEBPB'
#> ... Done loading TFBS(s) from TFregulomeR
#> Loading peak list y ... ...
#> ... You have 1 TFBS(s) requested to be loaded from TFregulomeR server
#> ... You chose to load TF peaks regardless of presence of motif. Using 'motif_only_for_id_y' tunes you
#> ... loading TFBS(s) from TFregulomeR now
#> ... ... peak file loaded successfully for id 'MM1_HSA_K562_ATF7'
#> ... Done loading TFBS(s) from TFregulomeR
#> Start analysing list x:MM1_HSA_K562_CEBPB... ...
#> ... ... Start analysing list y:MM1_HSA_K562_ATF7
#>
#> ....Assigning data to class slots!....
#>
#> ....Adding up PWMs!....
#>
#> ....PWMs are summed up and assigned as parent matrix!....
#>
#> ....Adding up Methylation Score matrices!....
#>
#>
#> Methylation Score Matrices are adding up....
#>
#>
#> ....Methylation Score Matrices are added and assigned as parent node's matrix....
#>
#> ....Merging matrices and converting them to Forked-TRANSFAC format!....
#>
#>
#> ... Matrices are converting to Forked-TRANSFAC format....
#>
#>
#> ...FPWM is stored at @forked slot of the object ....
#>
#> ....Methylation Score matrices are modified!....
#>
#> ....Parent node's Methylation Score Matrix is modified to right format!....
#>
#> ....Leaf nodes' Methylation Score Matrix is modified to right format!....
#>
#> ....Returning class object!....
```

The result output would be of the form follow:

```
AclassObject
#> An object of class "FPWMClassObj"
#> Slot "xid":
#> [1] "MM1_HSA_K562_CEBPB"
#>
#> Slot "id":
#> [[1]]
#> [1] "MM1_HSA_K562_CEBPD"
#>
#> [[2]]
#> [1] "MM1_HSA_K562_ATF4"
```

```
#>
#> [[3]]
#> [1] "MM1_HSA_K562_ATF7"
#>
#>
#> Slot "matrix":
#> [[1]]
#>          A    C    G    T
#>  [1,]  238  169  777  557
#>  [2,]  984  179  578    0
#>  [3,]    0    0    0 1741
#>  [4,]    0    0    0 1741
#>  [5,]   18    0 1617  106
#>  [6,]    0 1741    0    0
#>  [7,]  919   83  660   79
#>  [8,]  194  913    0  634
#>  [9,] 1022  719    0    0
#> [10,] 1741    0    0    0
#> [11,]    0  829  146  766
#>
#> [[2]]
#>          A    C    G    T
#>  [1,]  395  330 1503 1022
#>  [2,] 2175  295  780    0
#>  [3,]    0    0    0 3250
#>  [4,]    0    0    0 3250
#>  [5,]   43    0 2926  281
#>  [6,]    0 3250    0    0
#>  [7,] 2481  111  593   65
#>  [8,]  231  560    0 2459
#>  [9,]  596 2654    0    0
#> [10,] 3250    0    0    0
#> [11,]    0 1133  407 1710
#>
#> [[3]]
#>          A    C    G    T
#>  [1,]  216  188  827  534
#>  [2,] 1094  191  480    0
#>  [3,]    0    0    0 1765
#>  [4,]    0    0    0 1765
#>  [5,]   41    0 1553  171
#>  [6,]    0 1765    0    0
#>  [7,] 1325   82  323   35
#>  [8,]  130  317    0 1318
#>  [9,]  374 1391    0    0
#> [10,] 1765    0    0    0
#> [11,]    0  616  214  935
#>
#>
#> Slot "betalevel":
#> [[1]]
#>       [,1] [,2]  [,3]
#>  [1,] "6"  "133" "beta score < 10%"
#>  [2,] "7"  "0"   "beta score < 10%"
#>  [3,] "8"  "0"   "beta score < 10%"
#>  [4,] "9"  "0"   "beta score < 10%"
#>  [5,] "10" "0"   "beta score < 10%"
#>  [6,] "11" "7"   "beta score < 10%"
#>  [7,] "6"  "167" "beta score in between"
#>  [8,] "7"  "0"   "beta score in between"
#>  [9,] "8"  "0"   "beta score in between"
#> [10,] "9"  "0"   "beta score in between"
#> [11,] "10" "0"   "beta score in between"
#> [12,] "11" "0"   "beta score in between"
#> [13,] "6"  "104" "beta score > 90%"
#> [14,] "7"  "0"   "beta score > 90%"
#> [15,] "8"  "0"   "beta score > 90%"
#> [16,] "9"  "0"   "beta score > 90%"
#> [17,] "10" "0"   "beta score > 90%"
#> [18,] "11" "0"   "beta score > 90%"
#>
#> [[2]]
#>       [,1] [,2]  [,3]
#>  [1,] "6"  "161" "beta score < 10%"
#>  [2,] "7"  "0"   "beta score < 10%"
#>  [3,] "8"  "0"   "beta score < 10%"
#>  [4,] "9"  "0"   "beta score < 10%"
#>  [5,] "10" "0"   "beta score < 10%"
#>  [6,] "11" "15"  "beta score < 10%"
#>  [7,] "6"  "150" "beta score in between"
#>  [8,] "7"  "0"   "beta score in between"
#>  [9,] "8"  "0"   "beta score in between"
#> [10,] "9"  "0"   "beta score in between"
#> [11,] "10" "0"   "beta score in between"
#> [12,] "11" "8"   "beta score in between"
#> [13,] "6"  "44"  "beta score > 90%"
#> [14,] "7"  "0"   "beta score > 90%"
#> [15,] "8"  "0"   "beta score > 90%"
#> [16,] "9"  "0"   "beta score > 90%"
#> [17,] "10" "0"   "beta score > 90%"
#> [18,] "11" "0"   "beta score > 90%"
#>
#> [[3]]
#>       [,1] [,2]  [,3]
#>  [1,] "6"  "113" "beta score < 10%"
#>  [2,] "7"  "0"   "beta score < 10%"
#>  [3,] "8"  "0"   "beta score < 10%"
#>  [4,] "9"  "0"   "beta score < 10%"
#>  [5,] "10" "0"   "beta score < 10%"
#>  [6,] "11" "8"   "beta score < 10%"
#>  [7,] "6"  "55"  "beta score in between"
#>  [8,] "7"  "0"   "beta score in between"
#>  [9,] "8"  "0"   "beta score in between"
#> [10,] "9"  "0"   "beta score in between"
#> [11,] "10" "0"   "beta score in between"
#> [12,] "11" "3"   "beta score in between"
#> [13,] "6"  "1"   "beta score > 90%"
#> [14,] "7"  "0"   "beta score > 90%"
#> [15,] "8"  "0"   "beta score > 90%"
#> [16,] "9"  "0"   "beta score > 90%"
#> [17,] "10" "0"   "beta score > 90%"
#> [18,] "11" "0"   "beta score > 90%"
#>
#>
#> Slot "score":
#> [[1]]
#> [1] 19.16856
#>
#> [[2]]
#> [1] 37.36416
#>
#> [[3]]
#> [1] 20.63432
#>
#>
#> Slot "sp":
#> [1] 5
#>
#> Slot "parentmatrix":
#>         A   C    G    T
#> [1,]  849 687 3107 2113
#> [2,] 4253 665 1838    0
#> [3,]    0   0    0 6756
#> [4,]    0   0    0 6756
#> [5,]  102   0 6096  558
#>
#> Slot "parentbeta":
#>       [,1] [,2] [,3]
#>  [1,] "1"  "11" "beta score < 10%"
#>  [2,] "2"  "0"  "beta score < 10%"
#>  [3,] "3"  "0"  "beta score < 10%"
#>  [4,] "4"  "0"  "beta score < 10%"
#>  [5,] "5"  "0"  "beta score < 10%"
#>  [6,] "1"  "3"  "beta score in between"
#>  [7,] "2"  "0"  "beta score in between"
#>  [8,] "3"  "0"  "beta score in between"
#>  [9,] "4"  "0"  "beta score in between"
#> [10,] "5"  "0"  "beta score in between"
#> [11,] "1"  "0"  "beta score > 90%"
#> [12,] "2"  "0"  "beta score > 90%"
#> [13,] "3"  "0"  "beta score > 90%"
#> [14,] "4"  "0"  "beta score > 90%"
#> [15,] "5"  "0"  "beta score > 90%"
#>
#> Slot "forked":
#>    PO   V2   V3   V4   V5
#> 1   1  849  687 3107 2113
#> 2   2 4253  665 1838    0
#> 3   3    0    0    0 6756
#> 4   4    0    0    0 6756
#> 5   5  102    0 6096  558
#> 6   6    0 1741    0    0
#> 7    7  919   83  660   79
#> 8    8  194  913    0  634
#> 9    9 1022  719    0    0
#> 10  10 1741    0    0    0
#> 11  11    0  829  146  766
#> 12   6    0 3250    0    0
#> 13   7 2481  111  593   65
#> 14   8  231  560    0 2459
#> 15   9  596 2654    0    0
#> 16  10 3250    0    0    0
#> 17  11    0 1133  407 1710
#> 18   6    0 1765    0    0
#> 19   7 1325   82  323   35
#> 20   8  130  317    0 1318
#> 21   9  374 1391    0    0
#> 22  10 1765    0    0    0
#> 23  11    0  616  214  935
```

As it is shown, the output object, has multiple slots, each playing an essential role in plotting final figure. Motif and Beta Score matrices are forked regarding the user provided forking point. All the other neccessary information for figure, such as overlapping score, width/height and IDs are properly stored both for easier plotting and local storing. However, in order to generate a local file and store the whole data, it is necessary to define a standard format which is compatible with all current tools and applications. One of the widely used formats in this matter is TRANSFAC. The package TFregulomeR can provide the user with local data files of the TRANSFAC format. Below is shown a Position Weight Matrix in standard TRANSFAC format which is stored by exportMMPFM() function of TFregulomeR() package.

```
writeLines(readLines("MM1_HSA_K562_CEBPB_overlapped_with_MM1_HSA_K562_ATF4-motif-TRANSFAC.txt"))
#> AC MM1_HSA_K562_CEBPB_overlapped_with_MM1_HSA_K562_ATF4
#> XX
#> ID MM1_HSA_K562_CEBPB_overlapped_with_MM1_HSA_K562_ATF4
#> XX
#> DE MM1_HSA_K562_CEBPB_overlapped_with_MM1_HSA_K562_ATF4  ; from TFregulomeR
#> PO   A   C   G   T
#> 1    395 330 1503    1022
#> 2    2175    295 780 0
#> 3    0   0   0   3250
#> 4    0   0   0   3250
#> 5    43  0   2926    281
#> 6    0   3250    0   0
#> 7    2481    111 593 65
#> 8    231 560 0   2459
#> 9    596 2654    0   0
#> 10   3250    0   0   0
#> 11   0   1133    407 1710
#> XX
#> CC program: TFregulomeR
#> XX
#> //
```

0.3 TRANSFAC format and Forked-TRANSFAC

Note the format of the matrix, which is constructed by 5 columns: Position, A, C, G, T. Position column, holds the number of the location at which the frequency of each nucleotide is held. Since FPWM's approach needs to deal with multiple TFs instead of one, it merges all the matrices into one matrix, with a minor manipulation in the Positions column. Here is a Position Weight Matrix which is constructed by merging multiple PWMs.

```
as.matrix(AclassObject@forked)
#>    PO   V2   V3   V4   V5
#> 1   1  849  687 3107 2113
#> 2   2 4253  665 1838    0
#> 3   3    0    0    0 6756
#> 4   4    0    0    0 6756
#> 5   5  102    0 6096  558
#> 6   6    0 1741    0    0
#> 7   7  919   83  660   79
#> 8   8  194  913    0  634
#> 9   9 1022  719    0    0
#> 10 10 1741    0    0    0
#> 11 11    0  829  146  766
#> 12  6    0 3250    0    0
#> 13  7 2481  111  593   65
#> 14  8  231  560    0 2459
#> 15  9  596 2654    0    0
#> 16 10 3250    0    0    0
#> 17 11    0 1133  407 1710
#> 18  6    0 1765    0    0
#> 19  7 1325   82  323   35
#> 20  8  130  317    0 1318
#> 21  9  374 1391    0    0
#> 22 10 1765    0    0    0
#> 23 11    0  616  214  935
```

As it is depicted in this figure, the “forked” slot of the constructed object, holds a matrix that is quite similar to that from the standard TRANFAC format. However, as can be seen, there is repetition in the position column. This repetition implies the presence of multiple data frames of multiple TFs, that have been merged into one single data frame to be employed to generate a file of TRANSFAC format. As it was shown, the object was generated by providing two sets of IDs. ObjectGenerator(peak_id_x =“MM1_HSA_K562_CEBPB”, peak_id_y_list = list(“MM1_HSA_K562_CEBPD”,“MM1_HSA_K562_ATF4”,“MM1_HSA_K562_ATF7”), sp = 5) “MM1_HSA_K562_CEBPB”, being peak_id_x, will be used to construct three IntersecPeakMatrixs with each of IDs in peak_id_y_list. SP, or forking postion, is number of the position, from which on we need to fork our aggregated original matrix. The Aggregated matrix, is constructed by elementwise sum of three matrices. So up to position TheObject@sp (=5) we will have one single matrix which is consrtucted by MatrixAdder() function called within ObjectGenerator(). As the result, for positions from 1 to sp, one set of data frame for “MM1_HSA_K562_CEBPB” generated. Regarding this, for positions from sp+1 to the end, we will have three separate sets of data frames, for MM1_HSA_K562_CEBPD“,”MM1_HSA_K562_ATF4" and “MM1_HSA_K562_ATF7”. This being explained the repetition of position from 6 to 11 three times.

0.4 Storing a .txt file of Forked-TRANSFAC format

This Forked-PWM can be stroed in standard TRANSFAC fromat with the help of function StoreFTRANSFACFile() and by providing The generated object into it.

```
FPWM::StoreFTRANSFACFile(TheObject = AclassObject)
#> [1] "MM1_HSA_K562_CEBPB___3-FTRANSFAC.txt"
writeLines(readLines("MM1_HSA_K562_CEBPB_overlapped_with_MM1_HSA_K562_ATF4-motif-TRANSFAC.txt"))
#> AC MM1_HSA_K562_CEBPB_overlapped_with_MM1_HSA_K562_ATF4
#> XX
#> ID MM1_HSA_K562_CEBPB_overlapped_with_MM1_HSA_K562_ATF4
#> XX
#> DE MM1_HSA_K562_CEBPB_overlapped_with_MM1_HSA_K562_ATF4  ; from TFregulomeR
#> PO   A   C   G   T
#> 1    395 330 1503    1022
#> 2    2175    295 780 0
#> 3    0   0   0   3250
#> 4    0   0   0   3250
#> 5    43  0   2926    281
#> 6    0   3250    0   0
#> 7    2481    111 593 65
#> 8    231 560 0   2459
#> 9    596 2654    0   0
#> 10   3250    0   0   0
#> 11   0   1133    407 1710
#> XX
#> CC program: TFregulomeR
#> XX
#> //
```

The name of file indicates the peak_id_x and umber of TFs that has been intersected with it. Finally, the stored data file is of standard TRANSFAC format which can be used by all the applications and users that are compatible with TRANSFAC format as shown:

```
writeLines(readLines("MM1_HSA_K562_CEBPB___3-FTRANSFAC.txt"))
#> Parent logo : MM1_HSA_K562_CEBPB
#> XX
#> Leaf logos : MM1_HSA_K562_CEBPD,MM1_HSA_K562_ATF4,MM1_HSA_K562_ATF7
#> Overlapping scores : 19.168562041951,37.3641647712914,20.6343189284812
#> XX
#> Forking point : 5
#> PO   A   C   G   T
#> 1    849 687 3107    2113
#> 2    4253    665 1838    0
#> 3    0   0   0   6756
#> 4    0   0   0   6756
#> 5    102 0   6096    558
#> 6    0   1741    0   0
#> 7    919 83  660 79
#> 8    194 913 0   634
#> 9    1022    719 0   0
#> 10   1741    0   0   0
#> 11   0   829 146 766
#> 6    0   3250    0   0
#> 7    2481    111 593 65
#> 8    231 560 0   2459
#> 9    596 2654    0   0
#> 10   3250    0   0   0
#> 11   0   1133    407 1710
#> 6    0   1765    0   0
#> 7    1325    82  323 35
#> 8    130 317 0   1318
#> 9    374 1391    0   0
#> 10   1765    0   0   0
#> 11   0   616 214 935
#> XX
#> CC program: Forked-PWM
#> Source: intersectPeakMatrix of MM1_HSA_K562_CEBPB and 3 other matrices.
#> Matirx built by 77.17 % Of total peaks.
#> XX
#> //
```

0.5 Generating multiple FPWMs and storing them, with user provided input

StoreFTRANSFACFile() takes an object which is properly generated, and stores a local .txt file of Forked-TRANSFAC format. Though if the users need a local file to be stored without generating the class object first, they can use StoreFTRANSFACFile() function to store a local file. This function will generate a local .txt file which can be a concatenation of multiple data frames each essential for generating one FPWM plot. It takes list of IDs and Forking points from user and using them, exports matrices from TFregulomeR() database, then stores a local file as it is depicted below:

```
FPWM::StoreMultiTRANSFACFile(List_peak_id_x = list("MM1_HSA_K562_CEBPB", "MM1_HSA_K562_ATF4"),
    Listof_peak_id_y_list = list(list("MM1_HSA_K562_CEBPD", "MM1_HSA_K562_ATF4",
        "MM1_HSA_K562_ATF7"), list("MM1_HSA_K562_CEBPB", "MM1_HSA_K562_JUN",
        "MM1_HSA_K562_JUND", "MM1_HSA_K562_ATF7")), List_sp = list(5, 4))
#>
#> ....Accessing TFregulomeR!....
#> TFregulomeR::intersectPeakMatrix() starting ... ...
#> You chose NOT to profile the methylation levels in 200bp window around peak summits
#> Loading peak list x ... ...
#> ... You have 1 TFBS(s) requested to be loaded from TFregulomeR server
#> ... You chose to load TF peaks with motif only. Using 'motif_only_for_id_x' tunes your options
#> ... loading TFBS(s) from TFregulomeR now
#> ... ... peak file loaded successfully for id 'MM1_HSA_K562_CEBPB'
#> ... Done loading TFBS(s) from TFregulomeR
#> Loading peak list y ... ...
#> ... You have 1 TFBS(s) requested to be loaded from TFregulomeR server
#> ... You chose to load TF peaks regardless of presence of motif. Using 'motif_only_for_id_y' tunes you
#> ... loading TFBS(s) from TFregulomeR now
#> ... ... peak file loaded successfully for id 'MM1_HSA_K562_CEBPD'
#> ... Done loading TFBS(s) from TFregulomeR
#> Start analysing list x:MM1_HSA_K562_CEBPB... ...
#> ... ... Start analysing list y:MM1_HSA_K562_CEBPD
#>
#> ....Accessing TFregulomeR!....
#> TFregulomeR::intersectPeakMatrix() starting ... ...
#> You chose NOT to profile the methylation levels in 200bp window around peak summits
#> Loading peak list x ... ...
#> ... You have 1 TFBS(s) requested to be loaded from TFregulomeR server
#> ... You chose to load TF peaks with motif only. Using 'motif_only_for_id_x' tunes your options
#> ... loading TFBS(s) from TFregulomeR now
#> ... ... peak file loaded successfully for id 'MM1_HSA_K562_CEBPB'
#> ... Done loading TFBS(s) from TFregulomeR
#> Loading peak list y ... ...
#> ... You have 1 TFBS(s) requested to be loaded from TFregulomeR server
#> ... You chose to load TF peaks regardless of presence of motif. Using 'motif_only_for_id_y' tunes you
#> ... loading TFBS(s) from TFregulomeR now
#> ... ... peak file loaded successfully for id 'MM1_HSA_K562_ATF4'
#> ... Done loading TFBS(s) from TFregulomeR
#> Start analysing list x:MM1_HSA_K562_CEBPB... ...
#> ... ... Start analysing list y:MM1_HSA_K562_ATF4
```

```
#>
#> ....Accessing TFregulomeR!....
#> TFregulomeR::intersectPeakMatrix() starting ... ...
#> You chose NOT to profile the methylation levels in 200bp window around peak summits
#> Loading peak list x ... ...
#> ... You have 1 TFBS(s) requested to be loaded from TFregulomeR server
#> ... You chose to load TF peaks with motif only. Using 'motif_only_for_id_x' tunes your options
#> ... loading TFBS(s) from TFregulomeR now
#> ... ... peak file loaded successfully for id 'MM1_HSA_K562_CEBPB'
#> ... Done loading TFBS(s) from TFregulomeR
#> Loading peak list y ... ...
#> ... You have 1 TFBS(s) requested to be loaded from TFregulomeR server
#> ... You chose to load TF peaks regardless of presence of motif. Using 'motif_only_for_id_y' tunes you
#> ... loading TFBS(s) from TFregulomeR now
#> ... ... peak file loaded successfully for id 'MM1_HSA_K562_ATF7'
#> ... Done loading TFBS(s) from TFregulomeR
#> Start analysing list x:MM1_HSA_K562_CEBPB... ...
#> ... ... Start analysing list y:MM1_HSA_K562_ATF7
#>
#> ....Assigning data to class slots!....
#>
#> ....Adding up PWMs!....
#>
#> ....PWMs are summed up and assigned as parent matrix!....
#>
#> ....Adding up Methylation Score matrices!....
#>
#>
#> Methylation Score Matrices are adding up....
#>
#>
#> ....Methylation Score Matrices are added and assigned as parent node's matrix....
#>
#> ....Merging matrices and converting them to Forked-TRANSFAC format!....
#>
#>
#> ... Matrices are converting to Forked-TRANSFAC format....
#>
#>
#> ...FPWM is stored at @forked slot of the object ....
#>
#> ....Methylation Score matrices are modified!....
#>
#> ....Parent node's Methylation Score Matrix is modified to right format!....
#>
#> ....Leaf nodes' Methylation Score Matrix is modified to right format!....
#>
#> ....Returning class object!....
#>
#> ....Accessing TFregulomeR!....
#> TFregulomeR::intersectPeakMatrix() starting ... ...
#> You chose NOT to profile the methylation levels in 200bp window around peak summits
#> Loading peak list x ... ...
#> ... You have 1 TFBS(s) requested to be loaded from TFregulomeR server
#> ... You chose to load TF peaks with motif only. Using 'motif_only_for_id_x' tunes your options
#> ... loading TFBS(s) from TFregulomeR now
#> ... ... peak file loaded successfully for id 'MM1_HSA_K562_ATF4'
#> ... Done loading TFBS(s) from TFregulomeR
#> Loading peak list y ... ...
#> ... You have 1 TFBS(s) requested to be loaded from TFregulomeR server
#> ... You chose to load TF peaks regardless of presence of motif. Using 'motif_only_for_id_y' tunes you
#> ... loading TFBS(s) from TFregulomeR now
#> ... ... peak file loaded successfully for id 'MM1_HSA_K562_CEBPB'
#> ... Done loading TFBS(s) from TFregulomeR
#> Start analysing list x:MM1_HSA_K562_ATF4... ...
#> ... ... Start analysing list y:MM1_HSA_K562_CEBPB
#>
#> ....Accessing TFregulomeR!....
#> TFregulomeR::intersectPeakMatrix() starting ... ...
#> You chose NOT to profile the methylation levels in 200bp window around peak summits
#> Loading peak list x ... ...
#> ... You have 1 TFBS(s) requested to be loaded from TFregulomeR server
#> ... You chose to load TF peaks with motif only. Using 'motif_only_for_id_x' tunes your options
#> ... loading TFBS(s) from TFregulomeR now
#> ... ... peak file loaded successfully for id 'MM1_HSA_K562_ATF4'
#> ... Done loading TFBS(s) from TFregulomeR
#> Loading peak list y ... ...
#> ... You have 1 TFBS(s) requested to be loaded from TFregulomeR server
#> ... You chose to load TF peaks regardless of presence of motif. Using 'motif_only_for_id_y' tunes you
#> ... loading TFBS(s) from TFregulomeR now
#> ... ... peak file loaded successfully for id 'MM1_HSA_K562_JUN'
#> ... Done loading TFBS(s) from TFregulomeR
#> Start analysing list x:MM1_HSA_K562_ATF4... ...
#> ... ... Start analysing list y:MM1_HSA_K562_JUN
#>
#> ....Accessing TFregulomeR!....
#> TFregulomeR::intersectPeakMatrix() starting ... ...
#> You chose NOT to profile the methylation levels in 200bp window around peak summits
#> Loading peak list x ... ...
#> ... You have 1 TFBS(s) requested to be loaded from TFregulomeR server
#> ... You chose to load TF peaks with motif only. Using 'motif_only_for_id_x' tunes your options
#> ... loading TFBS(s) from TFregulomeR now
#> ... ... peak file loaded successfully for id 'MM1_HSA_K562_ATF4'
#> ... Done loading TFBS(s) from TFregulomeR
#> Loading peak list y ... ...
#> ... You have 1 TFBS(s) requested to be loaded from TFregulomeR server
#> ... You chose to load TF peaks regardless of presence of motif. Using 'motif_only_for_id_y' tunes you
#> ... loading TFBS(s) from TFregulomeR now
#> ... ... peak file loaded successfully for id 'MM1_HSA_K562_JUND'
#> ... Done loading TFBS(s) from TFregulomeR
#> Start analysing list x:MM1_HSA_K562_ATF4... ...
#> ... ... Start analysing list y:MM1_HSA_K562_JUND
#>
#> ....Accessing TFregulomeR!....
#> TFregulomeR::intersectPeakMatrix() starting ... ...
#> You chose NOT to profile the methylation levels in 200bp window around peak summits
#> Loading peak list x ... ...
#> ... You have 1 TFBS(s) requested to be loaded from TFregulomeR server
#> ... You chose to load TF peaks with motif only. Using 'motif_only_for_id_x' tunes your options
#> ... loading TFBS(s) from TFregulomeR now
#> ... ... peak file loaded successfully for id 'MM1_HSA_K562_ATF4'
#> ... Done loading TFBS(s) from TFregulomeR
#> Loading peak list y ... ...
#> ... You have 1 TFBS(s) requested to be loaded from TFregulomeR server
#> ... You chose to load TF peaks regardless of presence of motif. Using 'motif_only_for_id_y' tunes you
#> ... loading TFBS(s) from TFregulomeR now
#> ... ... peak file loaded successfully for id 'MM1_HSA_K562_ATF7'
#> ... Done loading TFBS(s) from TFregulomeR
#> Start analysing list x:MM1_HSA_K562_ATF4... ...
#> ... ... Start analysing list y:MM1_HSA_K562_ATF7
#>
#> ....Assigning data to class slots!....
#>
#> ....Adding up PWMs!....
#>
#> ....PWMs are summed up and assigned as parent matrix!....
#>
#> ....Adding up Methylation Score matrices!....
#>
#>
#> Methylation Score Matrices are adding up....
#>
#>
#> ....Methylation Score Matrices are added and assigned as parent node's matrix....
#>
#> ....Merging matrices and converting them to Forked-TRANSFAC format!....
#>
#>
#> ... Matrices are converting to Forked-TRANSFAC format....
#>
#>
#> ...FPWM is stored at @forked slot of the object ....
#>
#> ....Methylation Score matrices are modified!....
#>
#> ....Parent node's Methylation Score Matrix is modified to right format!....
#>
#> ....Leaf nodes' Methylation Score Matrix is modified to right format!....
#>
#> ....Returning class object!....
```

StoreMultiTRANSFACFile() takes three lists as input. “List_peak_id_x” is the list of peak_id_x that user is willing to generate multiple intersectPeakMatrixs with a list of peak_id_y. Basically, the length of List_peak_id_x indicates the number of plots or Class Objects that can be constructed. The second argument is “Listof_peak_id_y_list” which is a list of lists. each inner list holds a number of IDs that are supposed to be used as peak_id_y in intersecPeakMatrix() function. List_peak_id_x[i] will associate with the list at Listof_peak_id_y_list[i] to construct a FPWM. Finally, “List_sp()” is a list of forking positions for each set of FPWs. Needless to day, number of Forking positions is equal to number of IDs provided as a list in “List_peak_id_x”. The locally stored file, contains two set of FPWMs concatinaied together. First one is for List_peak_id_x[1] = “MM1_HSA_K562_CEBPB”, and the second one List_peak_id_x[2] = “MM1_HSA_K562_CEBPD”. Methylation Scores and IDs are clearly specified as it is shown below:

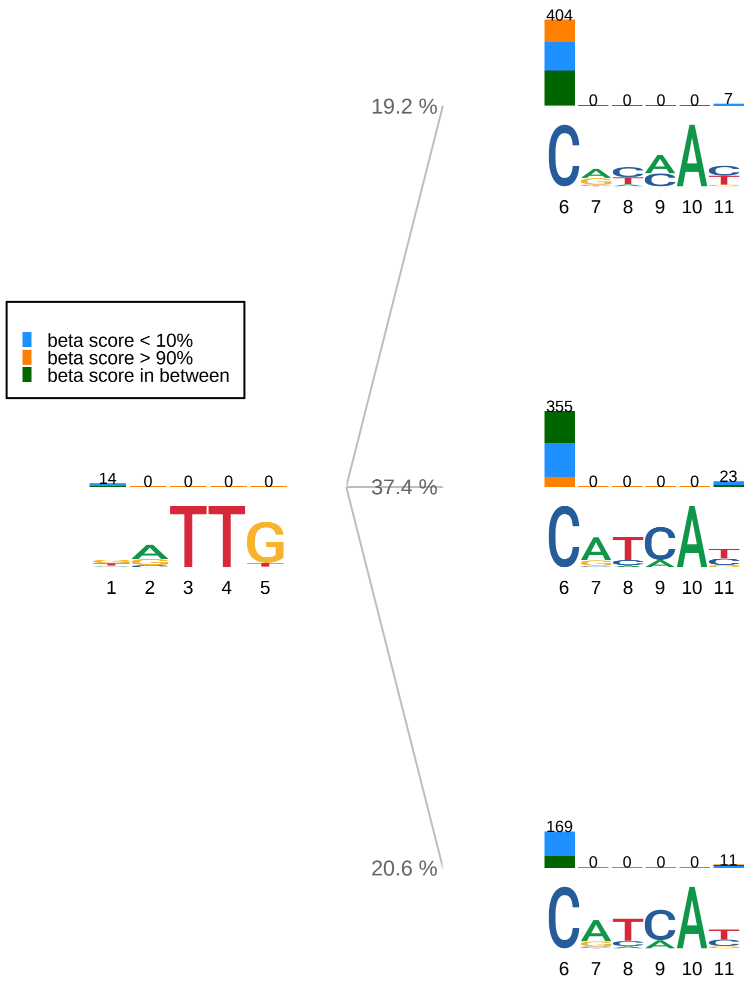
```
writeLines(readLines("All.txt"))
```

```
#> Parent logo : MM1_HSA_K562_CEBPB
#> XX
#> Leaf logos : MM1_HSA_K562_CEBPD,MM1_HSA_K562_ATF4,MM1_HSA_K562_ATF7
#> Overlapping scores : 19.168562041951,37.3641647712914,20.6343189284812
#> XX
#> Forking point : 5
#> PO   A   C   G   T
#> 1    849 687 3107    2113
#> 2    4253    665 1838    0
#> 3    0   0   0   6756
#> 4    0   0   0   6756
#> 5    102 0   6096    558
#> 6    0   1741    0   0
#> 7    919 83  660 79
#> 8    194 913 0   634
#> 9    1022    719 0   0
#> 10   1741    0   0   0
#> 11   0   829 146 766
#> 6    0   3250    0   0
#> 7    2481    111 593 65
#> 8    231 560 0   2459
#> 9    596 2654    0   0
#> 10   3250    0   0   0
#> 11   0   1133    407 1710
#> 6    0   1765    0   0
#> 7    1325    82  323 35
#> 8    130 317 0   1318
#> 9    374 1391    0   0
#> 10   1765    0   0   0
#> 11   0   616 214 935
#> XX
#> CC program: Forked-PWM
#> Source: intersectPeakMatrix of MM1_HSA_K562_CEBPB and 3 other matrices.
#> Matirx built by 77.17 % Of total peaks.
#> XX
#> //
#> Parent logo : MM1_HSA_K562_ATF4
#> XX
#> Leaf logos : MM1_HSA_K562_CEBPB,MM1_HSA_K562_JUN,MM1_HSA_K562_JUND,MM1_HSA_K562_ATF7
#> Overlapping scores : 16.6618329466357,7.68561484918794,13.8389404485692,12.4516627996906
#> XX
#> Forking point : 4
#> PO   A   C   G   T
#> 1    6631    1722    3000    166
#> 2    0   5   0   11514
#> 3    0   0   607 10912
#> 4    2182    0   7032    2305
#> 5    0   3779    0   1
#> 6    3116    172 492 0
#> 7    124 168 0   3488
#> 8    98  3677    0   5
#> 9    3775    0   2   3
#> 10   12  995 729 2044
#> 11   697 1442    476 1165
#> 5    0   1753    0   0
#> 6    1429    85  239 0
#> 7    33  79  0   1641
#> 8    27  1722    0   4
#> 9    1752    0   0   1
#> 10   8   394 419 932
#> 11   246 764 263 480
#> 5    0   3107    0   0
#> 6    2608    129 370 0
#> 7    54  135 0   2918
#> 8    47  3054    0   6
#> 9    3103    0   0   4
#> 10   11  666 722 1708
#> 11   468 1334    460 845
#> 5    0   2878    0   1
#> 6    2334    121 424 0
#> 7    41  86  0   2752
#> 8    46  2828    0   5
#> 9    2875    0   1   3
#> 10   9   613 571 1686
#> 11   421 1200    362 896
#> XX
#> CC program: Forked-PWM
#> Source: intersectPeakMatrix of MM1_HSA_K562_ATF4 and 4 other matrices.
#> Matirx built by 50.64 % Of total peaks.
#> XX
#> //
```

0.6 Plotting a FPWM class object

By providing the proper class object to FPWMPlotter(), the user can easily study a Forked-PWM constructed by multiple sequence logos, using PWM stored in the object. This function is also able to plot the beta level on the top of sequence logos if the Methylation argument is set on TRUE.

```
FPWM::FPWMPlotter(TheObject = AclassObject, Methylation = TRUE)
```



##Plotting a single local file of Forked-TRANSFAC format The ObjectGenerator() provides the user with the class object respecting user-provided inputs. Then the same object can be taken by FPWMplotter to deliver the final plot. However, if the user is willing to plot a local data file of Forked-TRANSFAC format, function ReadFTRANSFACFile() can be employed to generate a class object out of a local file.

```
library("stringr")

ObjectOfLocalFIle <- FPWM::ReadFTRANSFACFile(File = "MM1_HSA_K562_CEBPB___3-FTRANSFAC.txt",
    Methylation = FALSE)
```

Note that Methylation is set to FALSE. In case the user is willing to plot methylation score, thus the object contains the matrices, local matrices should be exported using TFregulomeR::exportMMPFM() to the same directory of Forked-TRANSFAC file. ReadFTRANSFACFile() will import Forked-TRANSFAC file first, and regarding the IDs provided within the .txt file, will look for associated Methylation Score file by browsing files names.

```
Peak_id_x <- ObjectOfLocalFIle@xid
Peak_id_y_List <- ObjectOfLocalFIle@id
```

Peak_id_x:

```
Peak_id_x
#> [1] "MM1_HSA_K562_CEBPB"
```

Peak_id_y_List:

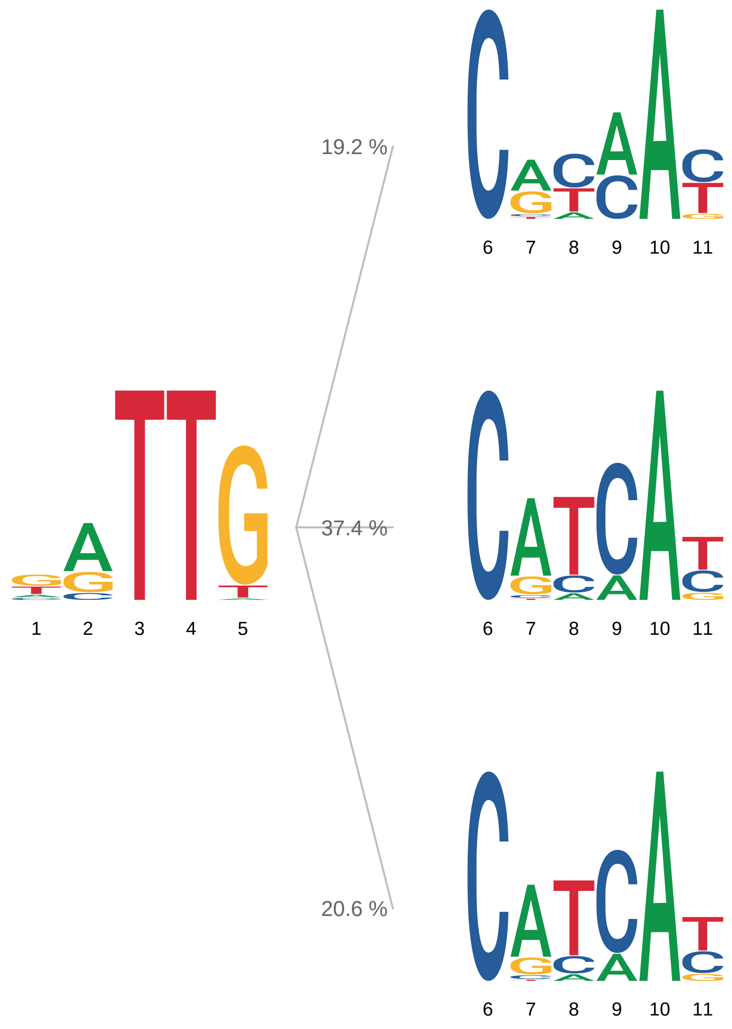
```
Peak_id_y_List
#> [[1]]
#> [1] "MM1_HSA_K562_CEBPD"
#>
#> [[2]]
#> [1] "MM1_HSA_K562_ATF4"
#>
#> [[3]]
#> [1] "MM1_HSA_K562_ATF7"
```

```
CEBPB_CEBPD <- TFregulomeR::intersectPeakMatrix(peak_id_x = "MM1_HSA_K562_CEBPB",
    motif_only_for_id_x = T, peak_id_y = "MM1_HSA_K562_CEBPD", motif_type = "TRANSFAC")
#> TFregulomeR::intersectPeakMatrix() starting ... ...
#> You chose NOT to profile the methylation levels in 200bp window around peak summits
#> Loading peak list x ... ...
#> ... You have 1 TFBS(s) requested to be loaded from TFregulomeR server
#> ... You chose to load TF peaks with motif only. Using 'motif_only_for_id_x' tunes your options
#> ... loading TFBS(s) from TFregulomeR now
#> ... ... peak file loaded successfully for id 'MM1_HSA_K562_CEBPB'
#> ... Done loading TFBS(s) from TFregulomeR
#> Loading peak list y ... ...
#> ... You have 1 TFBS(s) requested to be loaded from TFregulomeR server
#> ... You chose to load TF peaks regardless of presence of motif. Using 'motif_only_for_id_y' tunes you
#> ... loading TFBS(s) from TFregulomeR now
#> ... ... peak file loaded successfully for id 'MM1_HSA_K562_CEBPD'
#> ... Done loading TFBS(s) from TFregulomeR
#> Start analysing list x:MM1_HSA_K562_CEBPB... ...
#> ... ... Start analysing list y:MM1_HSA_K562_CEBPD

CEBPB_ATF4 <- TFregulomeR::intersectPeakMatrix(peak_id_x = "MM1_HSA_K562_CEBPB",
    motif_only_for_id_x = T, peak_id_y = "MM1_HSA_K562_ATF4", motif_type = "TRANSFAC")
#> TFregulomeR::intersectPeakMatrix() starting ... ...
#> You chose NOT to profile the methylation levels in 200bp window around peak summits
#> Loading peak list x ... ...
#> ... You have 1 TFBS(s) requested to be loaded from TFregulomeR server
#> ... You chose to load TF peaks with motif only. Using 'motif_only_for_id_x' tunes your options
#> ... loading TFBS(s) from TFregulomeR now
#> ... ... peak file loaded successfully for id 'MM1_HSA_K562_CEBPB'
#> ... Done loading TFBS(s) from TFregulomeR
#> Loading peak list y ... ...
#> ... You have 1 TFBS(s) requested to be loaded from TFregulomeR server
#> ... You chose to load TF peaks regardless of presence of motif. Using 'motif_only_for_id_y' tunes you
#> ... loading TFBS(s) from TFregulomeR now
#> ... ... peak file loaded successfully for id 'MM1_HSA_K562_ATF4'
#> ... Done loading TFBS(s) from TFregulomeR
#> Start analysing list x:MM1_HSA_K562_CEBPB... ...
#> ... ... Start analysing list y:MM1_HSA_K562_ATF4

CEBPB_ATF7 <- TFregulomeR::intersectPeakMatrix(peak_id_x = "MM1_HSA_K562_CEBPB",
    motif_only_for_id_x = T, peak_id_y = "MM1_HSA_K562_ATF7", motif_type = "TRANSFAC")
#> TFregulomeR::intersectPeakMatrix() starting ... ...
#> You chose NOT to profile the methylation levels in 200bp window around peak summits
#> Loading peak list x ... ...
#> ... You have 1 TFBS(s) requested to be loaded from TFregulomeR server
#> ... You chose to load TF peaks with motif only. Using 'motif_only_for_id_x' tunes your options
#> ... loading TFBS(s) from TFregulomeR now
#> ... ... peak file loaded successfully for id 'MM1_HSA_K562_CEBPB'
#> ... Done loading TFBS(s) from TFregulomeR
#> Loading peak list y ... ...
#> ... You have 1 TFBS(s) requested to be loaded from TFregulomeR server
#> ... You chose to load TF peaks regardless of presence of motif. Using 'motif_only_for_id_y' tunes you
#> ... loading TFBS(s) from TFregulomeR now
#> ... ... peak file loaded successfully for id 'MM1_HSA_K562_ATF7'
#> ... Done loading TFBS(s) from TFregulomeR
#> Start analysing list x:MM1_HSA_K562_CEBPB... ...
#> ... ... Start analysing list y:MM1_HSA_K562_ATF7

TFregulomeR::exportMMPFM(fun_output = CEBPB_CEBPD, fun = "intersectPeakMatrix",
    save_motif_PFM = T, save_betaScore_matrix = T)
#> Start exporting ... ...
#> ... ... You chose to save motif PFM and beta score matrix.
#> ... ... export intersectPeakMatrix...
#> ... ... we will export in the x wide of intersectPeakMatrix output since the input angle_of_matrix_fo
#> ... ... ... export id = MM1_HSA_K562_CEBPB
#> ... ... ... ... Beta score matrix has been saved as 'MM1_HSA_K562_CEBPB_overlapped_with_MM1_HSA_K562_
#> ... ... ... ... Motif PFM has been saved as 'MM1_HSA_K562_CEBPB_overlapped_with_MM1_HSA_K562_CEBPD-mo
TFregulomeR::exportMMPFM(fun_output = CEBPB_ATF4, fun = "intersectPeakMatrix",
    save_motif_PFM = T, save_betaScore_matrix = T)
#> Start exporting ... ...
#> ... ... You chose to save motif PFM and beta score matrix.
#> ... ... export intersectPeakMatrix...
#> ... ... we will export in the x wide of intersectPeakMatrix output since the input angle_of_matrix_fo
#> ... ... ... export id = MM1_HSA_K562_CEBPB
#> ... ... ... ... Beta score matrix has been saved as 'MM1_HSA_K562_CEBPB_overlapped_with_MM1_HSA_K562_
#> ... ... ... ... Motif PFM has been saved as 'MM1_HSA_K562_CEBPB_overlapped_with_MM1_HSA_K562_ATF4-mot
TFregulomeR::exportMMPFM(fun_output = CEBPB_ATF7, fun = "intersectPeakMatrix",
    save_motif_PFM = T, save_betaScore_matrix = T)
#> Start exporting ... ...
#> ... ... You chose to save motif PFM and beta score matrix.
#> ... ... export intersectPeakMatrix...
#> ... ... we will export in the x wide of intersectPeakMatrix output since the input angle_of_matrix_fo
#> ... ... ... export id = MM1_HSA_K562_CEBPB
#> ... ... ... ... Beta score matrix has been saved as 'MM1_HSA_K562_CEBPB_overlapped_with_MM1_HSA_K562_
#> ... ... ... ... Motif PFM has been saved as 'MM1_HSA_K562_CEBPB_overlapped_with_MM1_HSA_K562_ATF7-mot

library("stringr")
ObjectOfLocalFIle_withMethylation <- FPWM::ReadFTRANSFACFile(File = "MM1_HSA_K562_CEBPB___3-FTRANSFAC.tx
    Methylation = TRUE)
#>
#>
#> Methylation Score Matrices are adding up....
#>
#>
```

```
#> ....Methylation Score Matrices are added and assigned as parent node's matrix....
#>
#> ....Parent node's Methylation Score Matrix is modified to right format!....
#>
#> ....Leaf nodes' Methylation Score Matrix is modified to right format!....
```

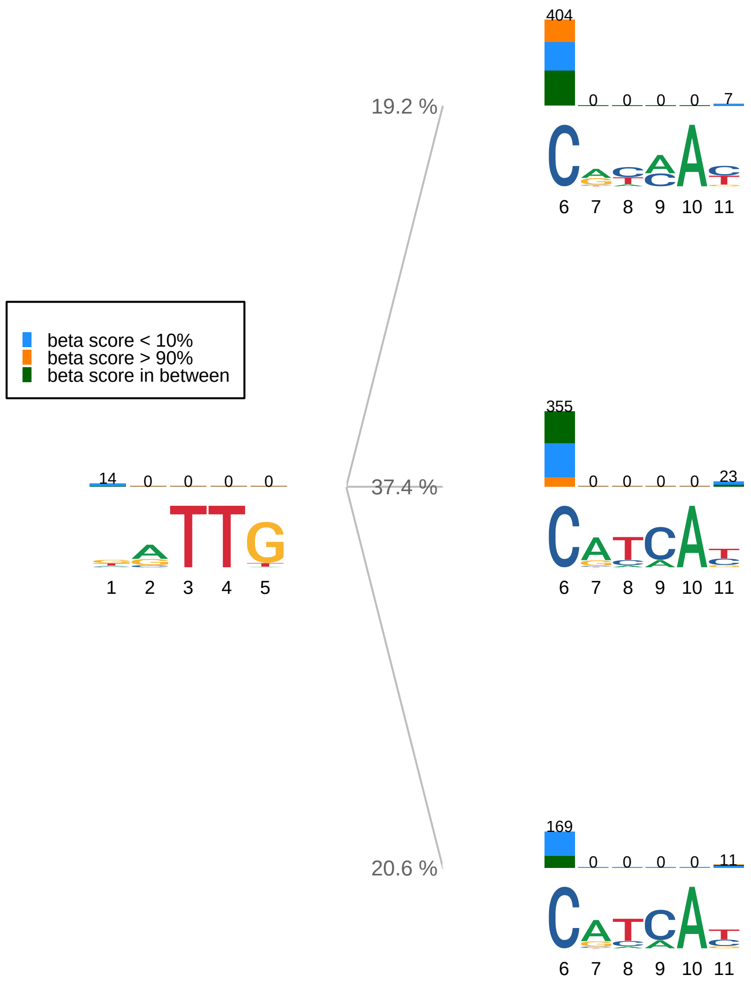
Now, with the object in hand, we can plot a FPWM with or without Beta levels. Note that Methylation argumetn of FPWMplotter() should be set accordingly. Without Methylation Scores locally available:

```
FPWM::FPWMPlotter(TheObject = ObjectOfLocalFIle, Methylation = FALSE)
```



With Methylation Score files locally available and ready to import:

```
FPWM::FPWMPlotter(TheObject = ObjectOfLocalFIle_withMethylation, Methylation = TRUE)
```



0.7 Storing plots by importing Bulk Data File

With previous approach, the user can observe a single FPWM plot only with the help of locally available Forked-TRANSFAC file. In addition to this, FPWM gives the user the ability to plot and store plenty of FPWMs by employing the function PlotMultiFTRANSFACFile(). This function takes the data file generated by FPWM::StoreMultiTRANSFACFile(), or of the same format and strcture, and stores PDF files of the final plot. It gives the user the ability to fastly browse big amount of data by feeding the only directory of the bulk data files.

```
library(stringr)
FPWM::PlotMultiFTRANSFACFile(File = "All.txt")
```