seqPattern: an R package for visualisation of oligonucleotide sequence patterns and motif occurrences

Vanja Haberle [1]

May 8, 2015

Contents

[1] vanja.haberle@gmail.com

1 Introduction

This document briefly describes how to use the package *seqPattern*. *seqPattern* is a Bioconductor-compliant R package designed to visualise sequence patterns across a large set of sequences (*e.g.* promoters, enhancers, ChIPseq peaks, *etc.*) centred at a common reference position (*e.g.* TSS, peak position) and ordered by some feature. The visualisations includes plotting the density of occurrences of di-, tri-, and in general any oligo-nucleotides, consensus sequences specified using IUPAC nucleotide ambiguity codes and motifs specified by a position weight matrix (PWM). Such visualisations are useful in discovering sequence patterns positionally constrained with respect to the reference point and their correlation with the specified feature [1].
Here is a list of functionalities provided in this package:

- Obtaining positions of occurrences of oligonucleotides or consensus sequences in an ordered set of sequences centred at a common reference point in a matrix-like representation.
- Visualising density of oligonucleotide or consensus sequence occurrences in an ordered set of sequences centred at a common reference point. Multiple oligonucleotides can be analysed simultaneously and their density plotted on the same colour scale allowing visual comparison of enrichments/depletions of various oligonucleotides.
- Visualising average oligonucleotide profile for a set of sequences centred at a common reference point.
- Obtaining positions of occurrences of motifs defined by a PWM in an ordered set of sequences centred at a common reference point in a matrix-like representation. A custom threshold can be applied to report only motif matches with score above specified percentage of the maximal PWM score.
- Visualising density of motif occurrences in an ordered set of sequences centred at a common reference point. Only motif matches with score above specified percentage of the maximal PWM score are visualised.
- Visualising motif scanning scores for an ordered set of sequences centred at a common reference point in the form of a heatmap.
- Visualising average motif occurrence profile (motif specified by a PWM) for a set of sequences centred at a common reference point.

2 Getting started

To load the *seqPattern* package into your R envirnoment type:

```
> library(seqPattern)
```

3 Example data

The package contains two data sets provided for illustrating the functionality of the package.

3.1 Zebrafish promoter sequences

The first dataset is `zebrafishPromoters` and can be loaded by typing:

```
> data(zebrafishPromoters)
> zebrafishPromoters

DNAStringSet object of length 1000:
       width seq
   [1]  1000 ATCACTGGGTGACAAGCTGTTATAACACGCTG...ATGGGTAATGTAAAAAATAATATGAAAAATGC
   [2]  1000 TGACAGAAATGTGGATGATGTGTGGATAATTG...TCCAGACAAGTAAGAGACCACCCCCTCAGAAA
   [3]  1000 GTAAATCTAGTTTTGGCTGTTCTCAGGGGTGT...TGAACAATAGAGCATATGAACTGAAAGATTTT
   [4]  1000 TTGTCACTATACACTGCCATTATATGAATCAC...TGTTACTTTTAGCCAGTGTCTGAGTAAGTTTA
   [5]  1000 CACACTTTATATAATGAACAAATAAATATATT...CGTTAGCATTAATGCTAGCTTATTTTCGGGGC
   ...   ... ...
 [996]  1000 AACTGATTGTATTTTAATGACATTACAACATC...AAAGTTTAAAGCCTTGAGGCTCTAAGGCCTTT
 [997]  1000 CGCAATGCTCAGTAAACTCTCTGAAACAGACA...TTAACTGTTTTAAACCTCAGAAGGAGTTTTCT
 [998]  1000 TTGAAAATAATAGGGGTGAATGTATGACATTT...TCAGTCTGATAGATGACGTCAGTGCTCTTCTT
 [999]  1000 GATGTTGTTTTTTGGCTCAAGAACTGAAGTAT...ACTGTATGCAATATTTAATGTGATGTATTTAC
[1000]  1000 TACACACTCTTGCACAAACTCGCATACACTAC...TGACCAGCCTGGTTTAAGCTGGGCTCCCAGCC

> head(zebrafishPromoters@elementMetadata)

DataFrame with 6 rows and 2 columns
        tpm interquantileWidth
  <numeric>          <numeric>
1   21.6422                 58
2  166.7262                 59
3   34.2380                 29
4   33.9379                191
5   34.1767                 52
6   76.3036                 10
```

It is a `DNAStringSet` object that contains sequence of 1000 randomly selected promoters active in zebrafish (*Danio rerio*) embryos at 24 hours past fertilisation (hpf). The data is taken from Nepal *et al.* [2], and represents regions flanking 400 bp upstream and 600 bp downstream of the dominant TSS detected by Cap analysis of gene expression (CAGE). In addition to genomic sequence, the object contains metadata providing CAGE tag per million values and interquantile width for each promoter. This small example dataset is intended to be used as input for running examples from *seqPattern* package help pages.

3.2 Zebrafish promoter coordinates and associated features

The second dataset is `zebrafishPromoters24h`, which can be loaded by typing:

```
> data(zebrafishPromoters24h)
> head(zebrafishPromoters24h)

   chr  start    end strand dominantTSS      tpm interquantileWidth
1 chr1  53498  53920      +       53734 58.10913                31
2 chr1 101877 102026      +      101934 43.38573                14
3 chr1 134648 134774      +      134725 80.04638                34
4 chr1 293528 293621      +      293616 54.59192                48
5 chr1 368940 369095      +      369041 55.50777                27
6 chr1 388651 388827      +      388696 10.14479                13
```

This is a `data.frame` object that contains genomic coordinates of all (10785 in total) promoters active in zebrafish *Danio rerio* embryos at 24 hpf [2]. For each promoter additional information is provided, including position of the dominant (most frequently used) TSS position, number of CAGE tags per million supporting that promoter and the interquantile width of the promoter (width of the central region containing >= 80% of the CAGE tags). All examples in this vignette use this dataset to demonstrate how to use various functions provided in the package and illustrate the resulting visualisation.

4 Visualising oligonucleotide and consensus sequence densities

In this part of the tutorial we will be using data from zebrafish (*Danio rerio*) that was mapped to the danRer7 assembly of the genome. Therefore, the corresponding genome package *BSgenome.Drerio.UCSC.danRer7* has to be installed and available to load by typing:

```
> library(BSgenome.Drerio.UCSC.danRer7)
```

4.1 Preparing input sequences

As input we will use a full set of zebrafish promoters active in 24 hpf embryos that were precisely mapped using CAGE [2]. To load the zebrafish promoters data type:

```
> data(zebrafishPromoters24h)
> nrow(zebrafishPromoters24h)

[1] 10785

> head(zebrafishPromoters24h)

   chr  start    end strand dominantTSS      tpm interquantileWidth
1 chr1  53498  53920      +       53734 58.10913                 31
2 chr1 101877 102026      +      101934 43.38573                 14
3 chr1 134648 134774      +      134725 80.04638                 34
4 chr1 293528 293621      +      293616 54.59192                 48
5 chr1 368940 369095      +      369041 55.50777                 27
6 chr1 388651 388827      +      388696 10.14479                 13
```

The loaded `data.frame` contains genomic coordinates, position of the dominant (most frequently used) TSS position, number of CAGE tags per million and the interquantile width (width of the central region containing $>=$ 80% of the CAGE tags) for each promoter.

Next, we need to obtain the genomic sequence of the promoter region for which the oligonucleotide density will be visualised, for instance the region flanking 400 bp upstream and 800 bp downstream of the dominant TSS. Thus, in this case the dominant TSS will be the reference point to which all promoter sequences will be aligned. We also want to keep the information about promoter interquantile width, since this feature will be used to order the promoters in the density map.

```
> # create GRanges of dominant TSS position with associated metadata
> zebrafishPromotersTSS <- GRanges(seqnames = zebrafishPromoters24h$chr,
+             ranges = IRanges(start = zebrafishPromoters24h$dominantTSS,
+             end = zebrafishPromoters24h$dominantTSS),
+             strand = zebrafishPromoters24h$strand,
+             interquantileWidth = zebrafishPromoters24h$interquantileWidth,
+             seqlengths = seqlengths(Drerio))
> # get regions flanking TSS - 400 bp upstream and 800 bp downstream
> zebrafishPromotersTSSflank <- promoters(zebrafishPromotersTSS, upstream = 400,
+                                         downstream = 800)
> # obtain genomic sequence of flanking regions
> zebrafishPromotersTSSflankSeq <- getSeq(Drerio, zebrafishPromotersTSSflank)
```

Note that all regions need to have the same width for plotting, and in cases when flanking regions fall outside of chromosome boundaries they need to be removed prior to plotting the oligonucleotide density map. (For instance, use the `trim` function from the *GenomicRanges* package to restrict regions within the boundaries of chromosomes and then remove ranges shorter than expected width.)

4.2 Plotting dinucleotide density maps

Once a `DNAStringSet` object is obtained, it can be used to plot the density of oligonucleotides of interest. In the following example, we will plot the density of AA, TA, CG and GC dinucleotides for the obtained set of sequences sorted by the promoter interquantile width (Figure 1):

```
> plotPatternDensityMap(regionsSeq = zebrafishPromotersTSSflankSeq,
+             patterns = c("AA", "TA", "CG", "GC"),
+             seqOrder = order(zebrafishPromotersTSSflank$interquantileWidth),
+             flankUp = 400, flankDown = 800, color = "blue")
```

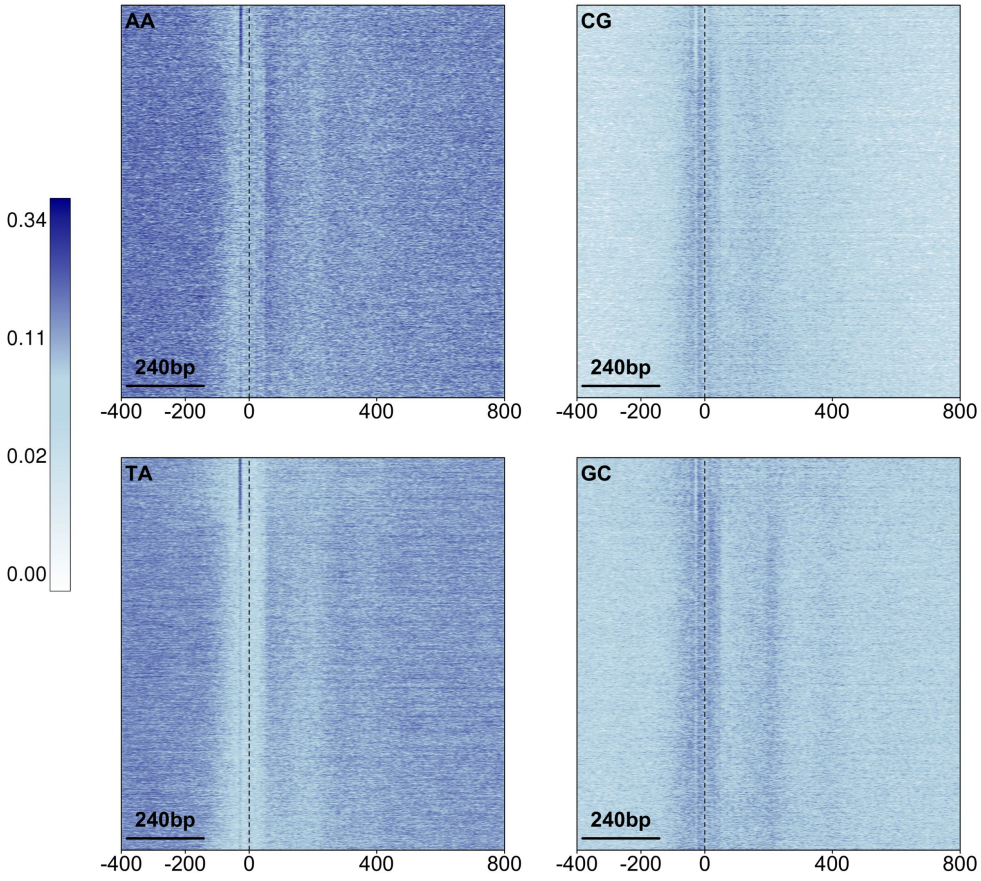

Figure 1: Density of AA, TA, CG and GC dinucleotides in regions flanking dominant TSS in zebrafish 24h embryo promoters ordered by promoter width

The information about the number of base pairs upstream and downstream of the reference point (`flankUp` and `flankDown`) needs to be specified in cases where these are asymmetric. If not specified, it is assumed that the reference point is located in the middle of the provided sequences (*i.e.* half of the total length in bp). There is also a number of graphical parameters that can be adjusted, such as color palette, width and hight of the plot in pixels, axis labels, scale bars, *etc.*, and they are explained in detail in the help page of the `plotPatternDensityMap` function. Default colour palette is in shades of blue going from white through lightblue to darkblue. Alternative colour palettes are available as shown in Figure 2, and can be specified by setting the `color` parameter to the name of the corresponding palette.

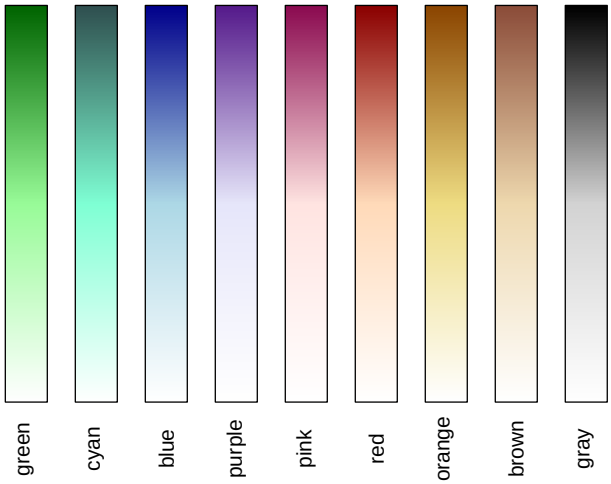

Figure 2: Available colour palettes for plotting oligonucleotide density maps. Name of each palette is indicated and can be used as `color` parameter within the `plotPatternDensityMap` function to specify the corresponding colour palette for plotting.

The two main parameters that define how the density of the pattern will be calculated and plotted are `nBin` and `bandWidth`. The `nBin` parameter specifies the number of bins in which the plot will be divided across x (horizontal, corresponding to the sequence length) and y axes (vertical, corresponding to the number of sequences). The default value is to calculate the density at each position in every sequence, *i.e.* the number of bins in the horizontal direction is set to the sequence length and in the vertical direction to the total number of sequences. The `bandWidth` parameter specifies the standard deviation of the bivariate Gaussian kernel along the x (*i.e.* number of nucleotides) and y (*i.e.* number of sequences) axis that is used to compute the density for each bin. The schematic in Figure 3 illustrates how the density of three different dinucleotides is calculated for a set of 10 sequences each 10 bp long, using a 2D Gaussian kernel with standard deviation of 5 in both directions. The calculations for larger sets of sequences and for any specified sequence pattern are done analogously. Note that *seqPattern* package supports parallel processing using multiple cores on Unix-like platforms, which significantly reduces the computational time when visualising density of multiple patterns. For instance, the above example that calculates the density of 4 dinucleotides can be run on 4 cores by setting the `useMulticore` and `nrCores` parameters:

```
> plotPatternDensityMap(regionsSeq = zebrafishPromotersTSSflankSeq,
+             patterns = c("AA", "TA", "CG", "GC"),
+             seqOrder = order(zebrafishPromotersTSSflank$interquantileWidth),
+             flankUp = 400, flankDown = 800, useMulticore = TRUE, nrCores = 4)
```

Calling the `plotPatternDensityMap` function will create one .png file per specified pattern in the working directory. The resulting dinucleotide density plots reveal complex pattern of dinucleotide enrichments/depletions in zebrafish promoters (Figure 1). The density of all

dinucleotides is plotted on the same colour scale, which allows direct comparison. For instance, it is clear that CG dinucleotides are generally less abundant than other dinucleotides. The region immediately upstream and downstream of TSS is enriched for CG and GC dinucleotides and depleted for TA and AA dinucleotides. Within this region, there is narrow band of TA and AA enrichment ~30 bp upstream of the TSS visible only in sharp promoters (top). Region downstream of TSS is characterised by alternating bands of enrichments and depletions visible for all four dinucleotides and this pattern is more prominent in broad promoters (bottom).Thus, visualising sequence in such way reveals differences in underlying sequence composition and its relation to the reference point, as well as how this changes with some associated feature.

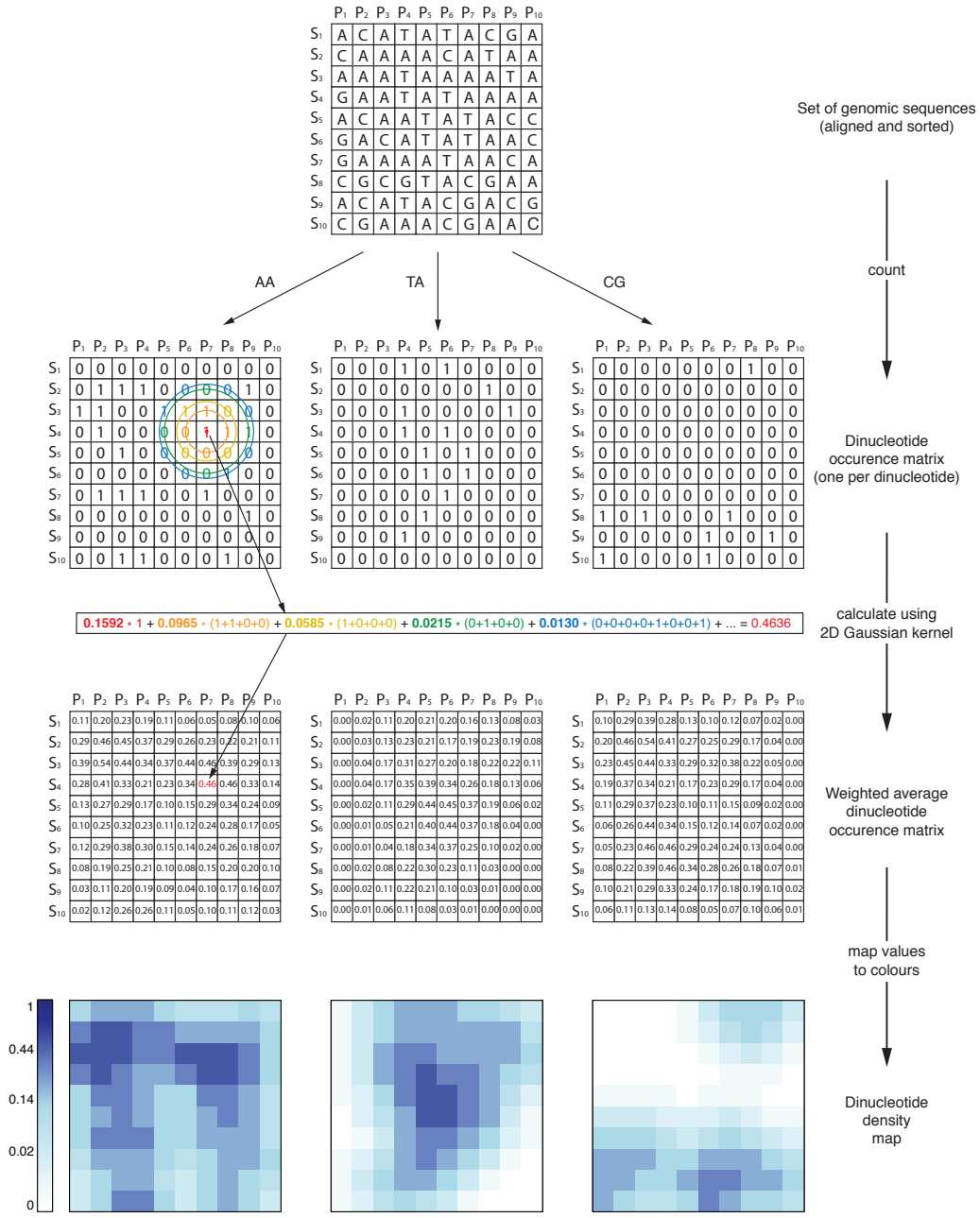

Figure 3: Schematics illustrating steps in pattern density calculation and visualisation. Genomic sequences (of the same length) are sorted and aligned into a matrix-like representation. Marking the presence of selected dinucleotide by 1 and the absence by 0 creates an occurrence matrix. Next, a weighted average is calculated at each position by placing a 2D Gaussian kernel at that position and assigning weights to surrounding positions. An example of calculating the value at position S4,P7 is shown. Surrounding positions are coloured on the basis of the weights assigned to them by the Gaussian kernel (bandwidth=5 in both dimensions, and covariance=0 between the two dimensions). Averaged values are mapped to different shades of blue to visualise the dinucleotide density.

In addition to plotting density map of individual dinucleotides, a "metadinucleotide" can be specified using IUPAC nucleotide ambiguity codes. For instance, for the same set of promoter regions we can plot the density of all WW and all SS dinucleotides in the same plot (Figure 4):

```
> plotPatternDensityMap(regionsSeq = zebrafishPromotersTSSflankSeq,
+             patterns = c("WW", "SS"),
+             seqOrder = order(zebrafishPromotersTSSflank$interquantileWidth),
+             flankUp = 400, flankDown = 800, color = "cyan", labelCol = "white")
```

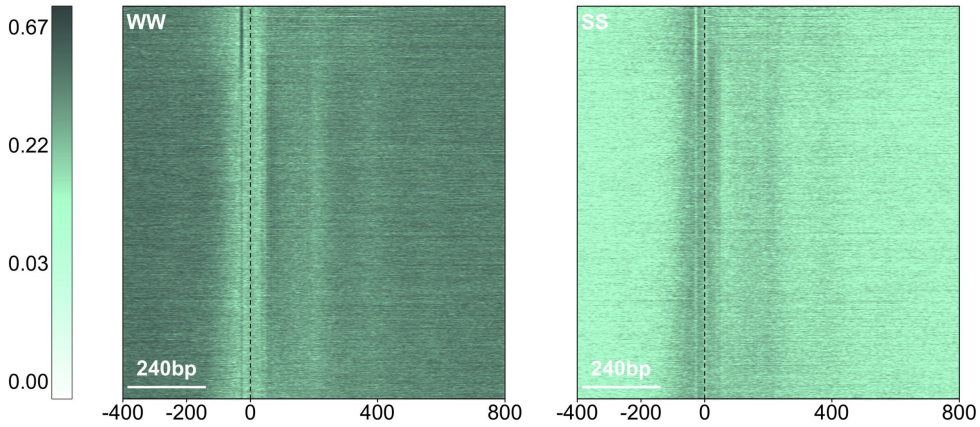

Figure 4: Density of WW and SS dinucleotides in regions flanking dominant TSS in zebrafish 24h embryo promoters ordered by promoter width

4.3 Plotting average oligonucleotide profiles

In addition to plotting oligonucleotide density maps, which show how a particular sequence pattern around referent point changes with some associated feature, the package also provides function for plotting average oligonucleotide profiles. For instance, we can visualise the average profile of WW and SS metadinucleotides for two groups of promoters separated by width (*i.e.* sharp and broad promoters; 5) as follows:

```
> # make index of sharp and broad promoters
> sIdx <- zebrafishPromotersTSSflank$interquantileWidth <= 9
> bIdx <- zebrafishPromotersTSSflank$interquantileWidth > 9
> # plot average dinucleotide profile for sharp promoters
> par(mfrow = c(1,2), mar = c(4.5,4,1,1))
> plotPatternOccurrenceAverage(regionsSeq = zebrafishPromotersTSSflankSeq[sIdx],
+             patterns = c("WW", "SS"), flankUp = 400, flankDown = 800,
+             smoothingWindow = 3, color = c("red3", "blue3"), cex.axis = 0.9)
```

```
> # plot average dinucleotide profile for broad promoters
> plotPatternOccurrenceAverage(regionsSeq = zebrafishPromotersTSSflankSeq[bIdx],
+             patterns = c("WW", "SS"), flankUp = 400, flankDown = 800,
+             smoothingWindow = 3, color = c("red3", "blue3"), cex.axis = 0.9)
```

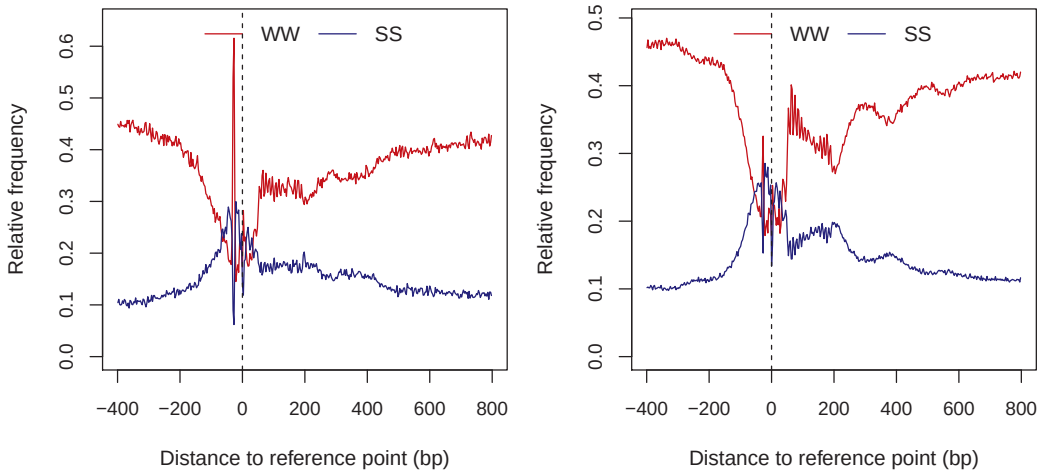

Figure 5: Average profile of WW and SS dinucleotides in region flanking dominant TSS for sharp promoters (width $<$ 10 bp; left) and broad promoters (width $>=$ 10 bp)

4.4 Plotting consensus sequence density map

Analogously to plotting dinucleotide density as described above, the `plotPatternDensityMap` function can be used to visualise the density of longer consensus sequences specified using IUPAC nucleotide ambiguity codes. For instance, one can use a consensus sequence of a transcription factor binding site to visualise density of these sites with respect to some reference point. Here we show an example of plotting density of the TATA-box consensus sequence (TATAWAWR) and GC-box / Sp1 binding site consensus sequence (RGGMGGR) across zebrafish promoters for sense and antisense strand separately. (Note that the easiest way to visualise occurrence of a consensus sequence in the antisense strand is to use the reverse complement of the consensus sequence, which in our case corresponds to YWTWTATA and YCCKCCY, for TATA-box and GC-box respectively.)

```
> # get regions flanking dominant TSS - 200bp upstream and downstream
> zebrafishPromotersTSSflank <- promoters(zebrafishPromotersTSS, upstream = 200,
+                                         downstream = 200)
> # obtain genomic sequence of the flanking regions
> zebrafishPromotersTSSflankSeq <- getSeq(Drerio, zebrafishPromotersTSSflank)
```

```
> # plot density of TATA-box consensus sequence in pink
> plotPatternDensityMap(regionsSeq = zebrafishPromotersTSSflankSeq,
+             patterns = c("TATAWAWR", "YWTWTATA"),
+             seqOrder = order(zebrafishPromotersTSSflank$interquantileWidth),
+             flankUp = 200, flankDown = 200, nBin = c(400, 10000),
+             bandWidth = c(1,6), color = "pink", addPatternLabel = FALSE)
>  # plot density of GC-box consensus sequence in purple
> plotPatternDensityMap(regionsSeq = zebrafishPromotersTSSflankSeq,
+             patterns = c("RGGMGGR", "YCCKCCY"),
+             seqOrder = order(zebrafishPromotersTSSflank$interquantileWidth),
+             flankUp = 200, flankDown = 200, nBin = c(400, 10000),
+             bandWidth = c(1,6), color = "purple", addPatternLabel = FALSE)
```

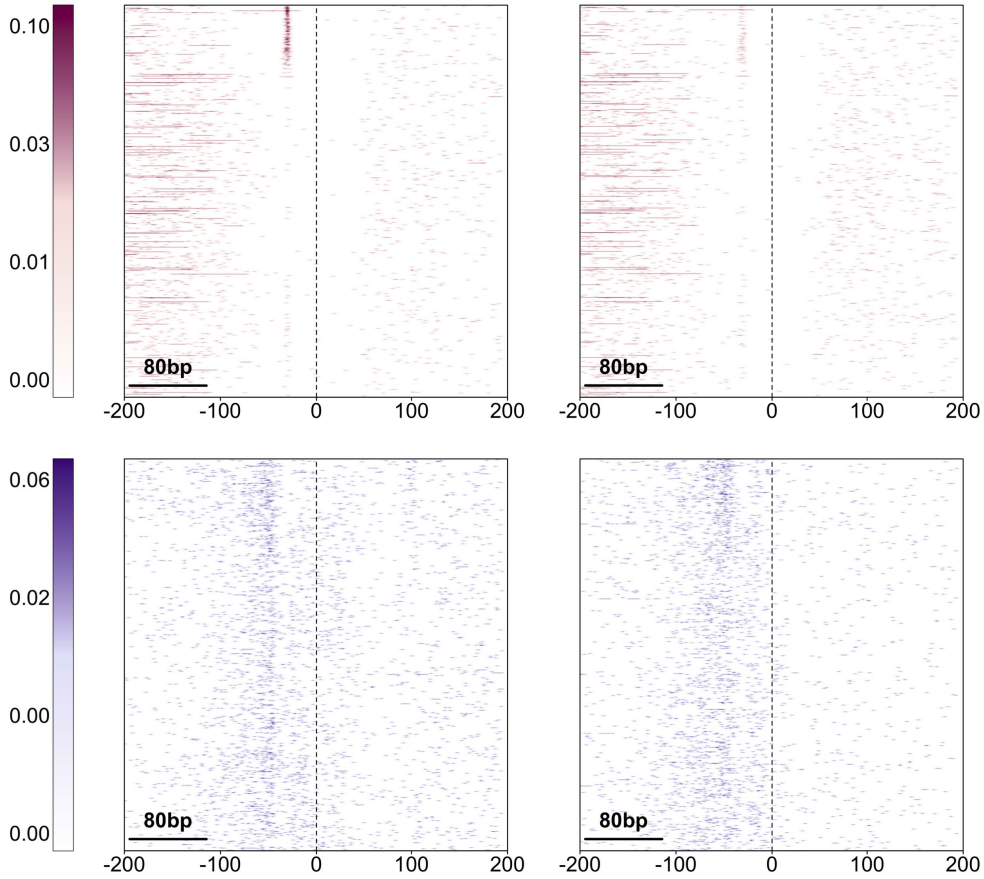

Figure 6: Density of the TATA-box consensus sequence (TATAWAWR; top) and GC-box (Sp1 binding site) consensus sequence (RGGMGGR; bottom) in regions flanking dominant TSS in zebrafish 24h embryo promoters ordered by promoter width. Matches in the sense strand are shown on the left and in the anti-sense strand on the right.

In the resulting density plot (Figure 6) promoters are aligned to dominant TSS and sorted by promoter width. This reveals that the TATA-box consensus sequence is positioned very precisely at ~30 bp upstream of the TSS mainly in the sense strand and that it is present only in very sharp promoters (top of the density plot), but not in the broad promoters (bottom of the density plot). In contrast, the GC-box (Sp1 binding site) consensus sequence is present in both strands and in both sharp and broad promoters roughly in the region 40-80 bp upstream of the TSS, but it seems more focused in sharp promoters.

5 Visualising motif occurrences

5.1 Plotting density of motif occurrences

In addition to using consensus sequence, the binding motif of a certain transcription factor can be described by a position-weight matrix (PWM), which gives the probability of occurrence of each of the four nucleotides at a given position in the motif. More specifically, the values in the PWM are derived from the position-specific frequency matrix and represent log-ratio between nucleotide probabilities derived from observed frequency and expected background probability for the corresponding nucleotide [3]. An example of a PWM describing the binding motif for the TATA-box binding transcription factor (TBP) is provided in the package and can be loaded:

```
> data(TBPpwm)
> TBPpwm

       [,1]      [,2]      [,3]      [,4]       [,5]      [,6]       [,7]
A -2.588668  1.853661 -4.928518  1.861830  1.4605751  1.886064  1.1891247
C -1.076769 -8.928518 -3.256093 -8.928518 -8.9285184 -4.928518 -5.4690868
G -2.420724 -5.469087 -5.469087 -4.228079 -8.9285184 -2.270307 -1.1406158
T  1.665806 -1.469087  1.941075 -1.690114  0.3146556 -3.974322  0.3146556
        [,8]
A  0.6713945
C -1.1406158
G  0.6898671
T -1.5534789
```

The `plotMotifDensityMap` function takes a PWM as an input, scans all sequences for the occurrence of the motif above the specified match threshold (*e.g.* 90%) and visualises the density of the motif with respect to the reference point (Figure 7, left):

```
> plotMotifDensityMap(regionsSeq = zebrafishPromotersTSSflankSeq,
+             motifPWM = TBPpwm, minScore = "90%",
+             seqOrder = order(zebrafishPromotersTSSflank$interquantileWidth),
```

```
+             flankUp = 200, flankDown = 200, color = "red")
```


5.2 Plotting motif scanning scores

On the other hand, using the `plotMotifScanScores` function it is possible to visualise the PWM scanning scores along entire sequences in a form of a heatmap (Figure 7, right):

```
> plotMotifScanScores(regionsSeq = zebrafishPromotersTSSflankSeq,
+             motifPWM = TBPpwm,
+             seqOrder = order(zebrafishPromotersTSSflank$interquantileWidth),
+             flankUp = 200, flankDown = 200)
```

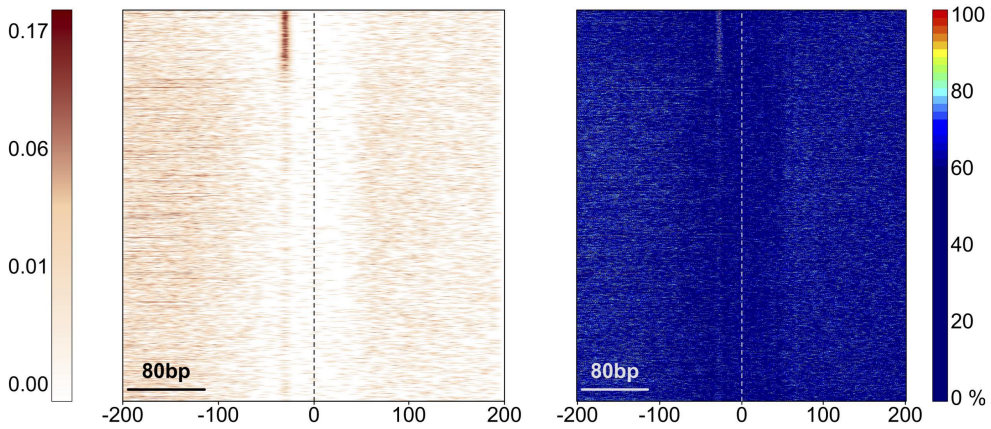

Figure 7: Density of TATA-box motif occurrence above 90% of PWM score (left) and heatmap of PWM scanning scores along all sequences (right) in regions flanking dominant TSS in zebrafish 24h embryo promoters ordered by promoter width

In addition to showing positioning and enrichment of strong motif occurrences with high match to the provided PWM, this form of visualisation can also reveal positional constraints for occurrences of motifs with varying strength. For instance weaker motifs might have different positional preference with respect to the reference point and might occur only in a subset of sequences correlating with some feature, which will be visible when the sequences are sorted according to that feature.

5.3 Plotting average motif occurrence profile

Analogously to plotting average oligonucleotide profiles described above, average occurrence profile of a motif specified by a PWM can be visualised using `plotMotifOccurrenceAverage` function. The following code exemplifies how to visualise average occurrence of a TATA-box motif (above 90% match to PWM) in previously defined subsets of sharp and broad promoters (Figure 8):

```
> # make index of sharp and broad promoters
> sIdx <- zebrafishPromotersTSSflank$interquantileWidth <= 9
> bIdx <- zebrafishPromotersTSSflank$interquantileWidth > 9
> # plot average motif occurrence profile for sharp promoters
> plotMotifOccurrenceAverage(regionsSeq = zebrafishPromotersTSSflankSeq[sIdx],
+             motifPWM = TBPpwm, minScore = "90%", flankUp = 200, flankDown = 200,
+             smoothingWindow = 3, color = c("red3"), cex.axis = 0.9)
> # add average motif occurrence profile for broad promoters to the existing plot
> plotMotifOccurrenceAverage(regionsSeq = zebrafishPromotersTSSflankSeq[bIdx],
+             motifPWM = TBPpwm, minScore = "90%", flankUp = 200, flankDown = 200,
+             smoothingWindow = 3, color = c("blue3"), add = TRUE)
> legend("topright", legend = c("sharp", "broad"), col = c("red3", "blue3"),
+ bty = "n", lwd = 1)
```

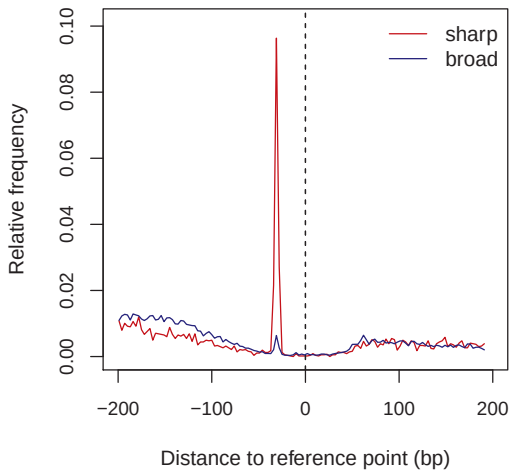

Figure 8: Average profile of TATA-box motif occurrence above 90% of PWM score in regions flanking dominant TSS in sharp (red) and broad (blue) promoters

6 Getting occurrence of sequence patterns and motifs without visualisation

The above described functions find the occurrence of specified sequence patterns or motifs in an ordered set of sequences, calculate their density and visualise the result as a density map. The *seqPattern* package also provides functions for finding only the occurrence of patterns or motifs without calculating the density and visualising it in a plot. These are `getPatternOccurrenceList` and `motifScanHits` for finding occurrence of patterns/consensus sequences and motifs specified by a PWM, respectively.

```
> motifOccurrence <- motifScanHits(regionsSeq =
+             zebrafishPromotersTSSflankSeq[1:50],
+             motifPWM = TBPpwm, minScore = "90%", seqOrder =
+             order(zebrafishPromotersTSSflank$interquantileWidth[1:50]))
> head(motifOccurrence)

  sequence position value
1        1       69     1
2        1      170     1
3        1      306     1
4        2       83     1
5        2      170     1
6        3      104     1
```

The occurrences are returned as coordinates in a matrix-like representation as follows: Input sequences of the same length are sorted according to the index in `seqOrder` argument creating an `n x m` matrix where `n` is the number of sequences and `m` is the length of the sequence. For each pattern match the coordinates within such matrix are reported, *i.e.* the ordinal number of the sequence within the ordered set of sequences (`sequence` column) and the start position of the pattern within that sequence (`position` column) are returned in the resulting `data.frame`.

Similarly, the matrix of PWM scanning scores along all sequences can be obtained using `motifScanScores` function:

```
> scanScores <- motifScanScores(regionsSeq = zebrafishPromotersTSSflankSeq[1:50],
+             motifPWM = TBPpwm, seqOrder =
+             order(zebrafishPromotersTSSflank$interquantileWidth[1:50]))
> dim(scanScores)

[1]  50 393

> scanScores[1:6,1:6]
```

```
         [,1]     [,2]     [,3]     [,4]     [,5]     [,6]
[1,] 63.43760 31.36077 57.53743 28.46517 35.59483 27.94817
[2,] 47.31515 57.84285 62.43671 67.52613 83.10196 54.81150
[3,] 71.92744 69.65404 82.63422 75.39365 71.82337 82.76337
[4,] 46.13987 62.04548 36.02750 46.60518 63.79247 49.47394
[5,] 90.57873 67.58995 90.99609 59.95292 79.81380 40.64799
[6,] 77.32850 66.54411 55.26975 39.65247 61.11497 44.48001
```

By default, the values corresponding to the percentage of the maximal possible PWM score are returned.

7 Session Info

```
> sessionInfo()

R version 4.6.0 (2026-04-24)
Platform: x86_64-pc-linux-gnu
Running under: Ubuntu 24.04.4 LTS

Matrix products: default
BLAS:   /usr/lib/x86_64-linux-gnu/openblas-pthread/libblas.so.3
LAPACK: /usr/lib/x86_64-linux-gnu/openblas-pthread/libopenblasp-r0.3.26.so;  LAPACK version 3

locale:
 [1] LC_CTYPE=C                 LC_NUMERIC=C
 [3] LC_TIME=C                  LC_COLLATE=C
 [5] LC_MONETARY=C              LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
 [9] LC_ADDRESS=C               LC_TELEPHONE=C
[11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C

time zone: Etc/UTC
tzcode source: system (glibc)

attached base packages:
[1] stats4    stats     graphics  grDevices utils     datasets  methods
[8] base

other attached packages:
 [1] BSgenome.Drerio.UCSC.danRer7_1.4.0 BSgenome_1.80.0
```

```
 [3] rtracklayer_1.72.0                BiocIO_1.22.0
 [5] Biostrings_2.80.1                 XVector_0.52.0
 [7] GenomicRanges_1.64.0              Seqinfo_1.2.0
 [9] IRanges_2.46.0                    S4Vectors_0.50.1
[11] BiocGenerics_0.58.1               generics_0.1.4
[13] seqPattern_1.44.0

loaded via a namespace (and not attached):
 [1] SparseArray_1.12.2          bitops_1.0-9
 [3] KernSmooth_2.23-26          lattice_0.22-9
 [5] digest_0.6.39               evaluate_1.0.5
 [7] grid_4.6.0                  fastmap_1.2.0
 [9] Matrix_1.7-5                cigarillo_1.2.0
[11] restfulr_0.0.16             BiocManager_1.30.27
[13] httr_1.4.8                  XML_3.99-0.23
[15] codetools_0.2-20            abind_1.4-8
[17] cli_3.6.6                   rlang_1.2.0
[19] crayon_1.5.3                BiocStyle_2.40.0
[21] Biobase_2.72.0              DelayedArray_0.38.2
[23] yaml_2.3.12                 plotrix_3.8-14
[25] S4Arrays_1.12.0             tools_4.6.0
[27] parallel_4.6.0              BiocParallel_1.46.0
[29] Rsamtools_2.28.0            SummarizedExperiment_1.42.0
[31] curl_7.1.0                  buildtools_1.0.0
[33] R6_2.6.1                    matrixStats_1.5.0
[35] xfun_0.57                   GenomicAlignments_1.48.0
[37] sys_3.4.3                   MatrixGenerics_1.24.0
[39] knitr_1.51                  rjson_0.2.23
[41] htmltools_0.5.9             rmarkdown_2.31
[43] maketools_1.3.2             compiler_4.6.0
[45] RCurl_1.98-1.18
```

References

[1] Vanja Haberle, Nan Li, Yavor Hadzhiev, Charles Plessy, Christopher Previti, Chirag Nepal, Jochen Gehrig, Xianjun Dong, Altuna Akalin, Ana Maria Suzuki, Wilfred F J van IJcken, Olivier Armant, Marco Ferg, Uwe Strähle, Piero Carninci, Ferenc Müller, and Boris Lenhard. Two independent transcription initiation codes overlap on vertebrate core promoters. *Nature*, 507(7492):381–385, 2014.

[2] Chirag Nepal, Yavor Hadzhiev, Christopher Previti, Vanja Haberle, Nan Li, Hazuki Takahashi, Ana Maria S. Suzuki, Ying Sheng, Rehab F. Abdelhamid, Santosh Anand, Jochen Gehrig, Altuna Akalin, Christel E.M. Kockx, Antoine A.J. van der Sloot, Wilfred F.J. van IJcken, Olivier Armant, Sepand Rastegar, Craig Watson, Uwe Strähle, Elia Stupka, Piero Carninci, Boris Lenhard, and Ferenc Müller. Dynamic regulation of coding and non-coding transcription initiation landscape at single nucleotide resolution during vertebrate embryogenesis. *Genome Research*, 23(11):1938–1950, 2013.

[3] Wyeth W Wasserman and Albin Sandelin. Applied bioinformatics for the identification of regulatory elements. *Nature Reviews Genetics*, 5(4):276–287, 2004.