Introduction to sequence motifs

Benjamin Jean-Marie Tremblay*

5 May 2026

Abstract

There are four ways to represent sequence motif matrices: as counts, probabilities, logodds scores, or information content. This vignette discusses the relationship between these and how they are obtained.

Contents

1 Introduction to sequence motifs **1**

2 Position count matrices **1**

3 Position probability matrices **2**

4 Position weight matrices **3**

5 Information content matrices **4**

6 Contribution weight matrices **8**

References **8**

1 Introduction to sequence motifs

Motifs are a more practical representation of consensus elements in biological sequences, allowing for a more detailed description of the variability at each site (see G. D. Stormo 2000). The `universalmotif` class supports Position Count Matrices (PCM), Position Probability Matrices (PPM), Position Weight Matrices (PWM), and Information Content Matrices (ICM), as well as Contribution Weight Matrices (CWM) from attribution analyses. Naming conventions vary between tools; these are the names used throughout this package.

The four conventional representations are described first, followed by CWMs and their different interpretation. Conversion helpers are documented in `?'utils-motif'` and `?convert_type`. The types are also reviewed in the motif manipulation vignette.

2 Position count matrices

Also known as position frequency matrices, these are typically the most basic representation of a motif. Simply, for each position the total counts of letters are shown. For example, the following sequences:

*benjamin.tremblay@uwaterloo.ca

Table 1: Starting sequences.

#	Sequence
1	AAGAAT
2	ATCATA
3	AAGTAA
4	AACAAA
5	ATTAAA
6	AAGAAT

... would be represented as:

Table 2: Position Count Matrix.

Position	1	2	3	4	5	6
A	6	4	0	5	5	4
C	0	0	2	0	0	0
G	0	0	3	0	0	0
T	0	2	1	1	1	2

(Note that all positions add up to 6, the initial number of sequences.)

3 Position probability matrices

Also sometimes known as position frequency matrices or position weight matrices, these represent the probabilities for each letter at each position. Using the previous motif (table 2) as an example, the following formula would be used to calculate the probability (P) of each letter N at an individual position from counts (C):

$$P(N) = \frac{C_N}{\sum C} \tag{1}$$

The equivalent vectorised `R` code:

```
PPM <- function(C) C / sum(C)
```

This leads to the following motif representation:

Table 3: Position Probability Matrix.

Position	1	2	3	4	5	6
A	1.00	0.67	0.00	0.83	0.83	0.66
C	0.00	0.00	0.33	0.00	0.00	0.00
G	0.00	0.00	0.50	0.00	0.00	0.00
T	0.00	0.33	0.17	0.17	0.17	0.33

(Note that all positions sum to 1.)

From this type of representation, the probability of any combination of letters can be calculated. For example, the probability for AAGAAA is about 15%. However, when starting from a small pool of sequences, many zeroes can appear in the PPM. This means that, for example, the probability of AAAAAA is currently zero.

When scanning through large numbers of biological sequences, throwing away combinations of letters such as these can potentially be undesirable, as sometimes mismatches can occur in some transcription factor binding sites. This can be fixed by adding a pseudocount (p). Usually a small number such as 1, it is introduced into the PCM to PPM calculation as such:

$$P_{\text{pseudo}}(N) = \frac{C_N + \frac{p}{n}}{\sum C + p} \tag{2}$$

The equivalent vectorised `R` code:

```
PPMp <- function(C, p) (C + p / length(C)) / (sum(C) + p)
```

In this equation, the pseudocount is added to the top and bottom of the fraction. However for the top fraction, which is specific to each letter, the pseudocount is divided by the total number of letters n (in the case of DNA, 4). This then generates the following motif:

Table 4: Position Probability Matrix with a pseudocount of 1.

Position	1	2	3	4	5	6
A	0.892	0.610	0.036	0.750	0.750	0.610
C	0.036	0.035	0.320	0.035	0.035	0.035
G	0.036	0.035	0.464	0.035	0.035	0.035
T	0.036	0.320	0.180	0.180	0.180	0.320

Now, though unlikely, it is no longer considered impossible for the sequence AAAAAA to exist as part of this motif. Since the total number of sequences in this case is quite low, the pseudocount can have a large impact on the values within the matrix. This impact decreases as the number of sequences increases. Of course, this can also be changed by using different pseudocounts.

4 Position weight matrices

The position weight matrix, also known as position-specific weight matrix, position-specific scoring matrix, and logodds scoring matrix, was first proposed by Gary D. Stormo et al. (1982). In this case for each position, every letter has a 'score' which can be used to evaluate how well a sequence matches a motif. Though there can be multiple ways of calculating these scores (S), the most common method is to calculate the log of each probability, correcting for background frequencies (B). This results in the following calculation:

$$S(N) = \log_2 \frac{P(N)}{B_N} \tag{3}$$

The equivalent vectorised `R` code:

```
S <- function(C, B) log2(PPM(C) / B)
```

Using this equation, fractions where the probability of a certain letter in a sequence is higher than the background probability of that letter result in positive scores after taking the log, and vice versa for negative scores. Using the table 3 motif and assuming a uniform background frequency (i.e. the probability of each of the four letters is 0.25), this results in the following PWM:

Table 5: Position Weight Matrix.

Position	1	2	3	4	5	6
A	2	1.425	-Inf	1.737	1.737	1.415
C	-Inf	-Inf	0.415	-Inf	-Inf	-Inf
G	-Inf	-Inf	1.000	-Inf	-Inf	-Inf
T	-Inf	0.415	-0.585	-0.585	-0.595	0.415

(Note that the position totals no longer have equal sums.)

In order to score a sequence, add up the score for the letters at the specific positions. For example AAGAAA has a score of 9.31. However, similar to with PPMs, if starting from a small pool of sequences the sequence AAAAAA could never be recovered using this motif model, with a score of -Inf. This can be avoided simply by starting from a pseudocount-adjusted PPM. Using the table 4 motif, this becomes:

Table 6: Position Weight Matrix with a pseudocount of 1.

Position	1	2	3	4	5	6
A	1.840	1.280	-2.807	1.585	1.585	1.280
C	-2.807	-2.807	0.363	-2.807	-2.807	-2.807
G	-2.807	-2.807	0.893	-2.807	-2.807	-2.807
T	-2.807	0.363	-0.485	-0.485	-0.485	0.363

Now, the score for AAGAAA is 8.46, and the score for AAAAAA is 4.76. Though the score for the latter is low, it is no longer -Inf as a result of one mismatch.

When searching for instances of this motif using this scoring system, one more consideration is needed: a minimum score. For example, both sequences AAGAAA and ATTTTT have positive scores; but one is much higher than the other. Should ATTTTT then be discarded? In order to answer this a threshold is set, typically this is a certain percent of the highest possible score. For example, in the table 6 motif, the highest possible score is 8.46. Using a threshold of 25%, the minimum score then becomes 2.115. This means that ATTTTT, with a score of 1.112, would indeed be discarded.

Another method for determining the minimum score includes starting from P-values. The package `TFMPvalue` for example can calculate minimum scores from P-values, using the algorithm described by Touzet and Varre (2007). The `universalmotif` package also offers this capability with the `motif_pvalue()` function. See the motif comparisons and P-values vignette for a detailed overview.

Whichever method you use however, it is important to remember that there is likely never one single threshold that works in every situation. Sequences passing a lower threshold can still be bound by transcription factors that bind DNA with less specificity or are present in higher concentrations, and the opposite can be true of higher-specificity or less concentrated transcription factors. See Djordjevic, Sengupta, and Shraiman (2003) and Ruan and Stormo (2017) for some background on the topic.

5 Information content matrices

Finally, the information content matrix (Thomas D. Schneider et al. 1986; T. D. Schneider and Stephens 1990). This type aims to include another consideration: are some positions more important than others? To explore this, let us consider the table 3 motif. This matrix can be represented as a probability sequence logo as seen in figure 1.

In this case, the height of the letters represents their probabilities at each position. However, when represented as an ICM, the sequence logo appears as seen in figure 2.

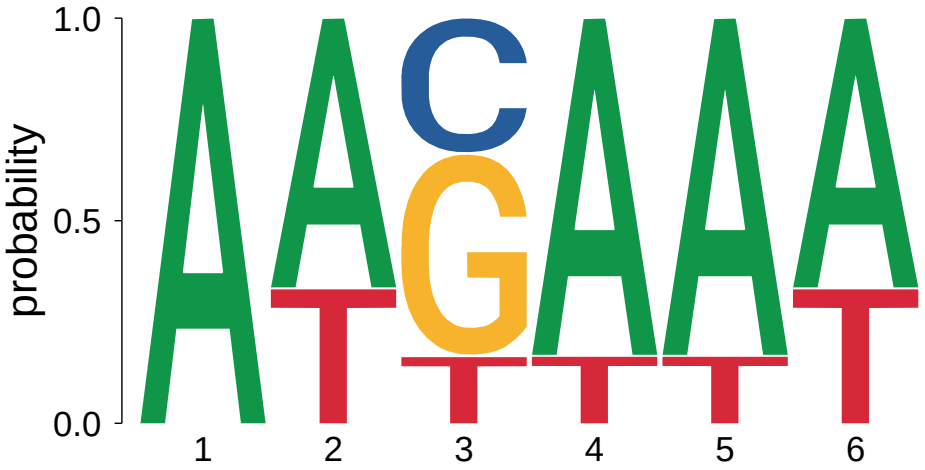

Figure 1: Sequence logo of a Position Probability Matrix

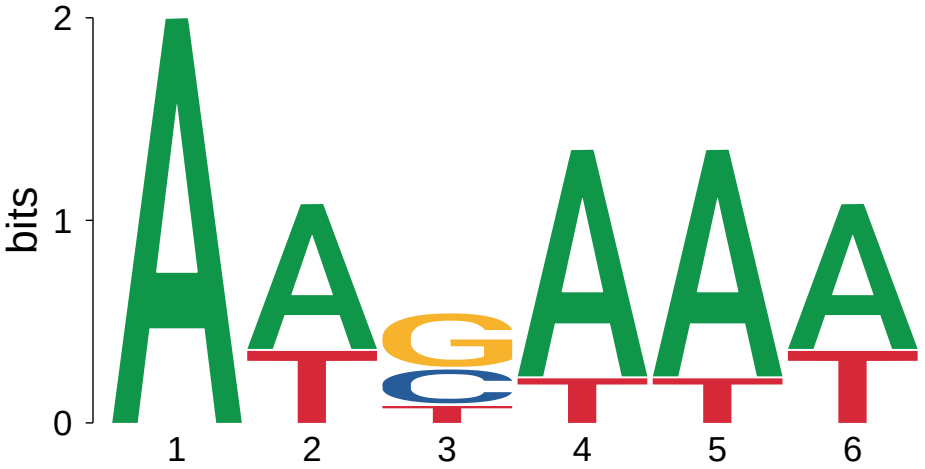

Figure 2: Sequence logo of an Information Content Matrix

Now, the total height of each position is scaled using the total *information* at that position. Simply put, the total information of each position is an indication of the level of conservation. In the example motif, the first position is highly conserved, always being the letter A, whereas the third position is less conserved, as the probabilities for any one letter are notably lower.

For every position, the letter heights are calculated by taking the total possible information content (IC), or maximum uncertainty, and then subtracting the actual uncertainty at each position (Thomas D. Schneider et al. 1986; T. D. Schneider and Stephens 1990). This calculation is based on Shannon's entropy (Shannon 1948), with the final values representing 'bits' (T. D. Schneider 1991). The total IC is based on alphabet length (n), using the following equation:

$$IC_{\text{total}} = \log_2 n \tag{4}$$

The equivalent `R` code:

```
ICtotal <- function(C) log2(length(C))
```

For DNA motifs which have an alphabet length of 4, the total IC is 2. To calculate the actual uncertainty (U) per position, the following equation is used:

$$U = -\sum_{N} P(N) \times \log_2 P(N) \tag{5}$$

The equivalent vectorised `R` code:

```
U <- function(C) -sum(PPM(C) * log2(PPM(C)), na.rm = TRUE)
```

Here, the position uncertainty is the sum of the uncertainties contributed by all alphabet letters (A, C, G, and T). To calculate the final information content:

$$IC_{\text{final}} = IC_{\text{total}} - U \tag{6}$$

The equivalent vectorised `R` code:

```
ICfinal <- function(C) ICtotal(C) - U(C)
```

In the original implementation described by Thomas D. Schneider et al. (1986), an additional error correction factor (E) is included to account for sample size. This correction is rarely used, but can be incorporated as follows:

$$IC_{\text{final}} = IC_{\text{total}} - [U + E] \tag{7}$$

The details for calculating this factor will not be covered here. (Refer to Thomas D. Schneider et al. (1986).) The `TFBSTools` package offers the ability to incorporate this error correction (and can be used in the `universalmotif` package via `convert_type(..., nsize_correction=TRUE)`).

Finally, to get the height of each letter at each position, the final IC is multiplied by the letter and position probabilities:

$$IC(N) = P(N) \times IC_{\text{final}} \tag{8}$$

The equivalent vectorised `R` code:

```
IC <- function(C) PPM(C) * ICfinal(C)
```

Using the above equations, the table 3 motif then becomes:

Table 7: Information Content Matrix.

Position	1	2	3	4	5	6
A	2.000	0.721	0.000	1.125	1.125	0.721
C	0.000	0.000	0.180	0.000	0.000	0.000
G	0.000	0.000	0.270	0.000	0.000	0.000
T	0.000	0.361	0.090	0.225	0.225	0.361

(Note that none of the positions have a sum larger than 2.)

An alternative to representing information content as Shannon's entropy is relative entropy, or Kullback-Leibler divergence (Kullback and Leibler 1951). While IC computed via Shannon's entropy has the advantage of having a consistent maximum IC for every position, it does not take into account non-uniform background frequencies. Relative entropy on the other hand will take this into account, but all positions no longer share the same maximum IC. To calculate relative entropy:

$$IC(N) = P(N) \times \log_2 \frac{P(N)}{B_N} \tag{9}$$

The equivalent vectorised `R` code:

```
IC <- function(C, B) pmax(PPM(C) * log2(PPM(C) / B), 0)
```

Using this equation can lead to IC less than zero. These values are not allowed, so they are simply replaced with zero. With this equation and assuming uniform background frequencies, the table 3 motif becomes:

Table 8: Information Content Matrix as relative divergence.

Position	1	2	3	4	5	6
A	1.640	0.777	0.000	1.190	1.190	0.777
C	0.000	0.000	0.177	0.000	0.000	0.000
G	0.000	0.000	0.415	0.000	0.000	0.000
T	0.000	0.177	0.090	0.000	0.000	0.117

This motif would look significantly different with non-uniform background frequencies. For example, starting from the following background frequencies: `c(A = 0.4, C = 0.1, G = 0.1, T = 0.4)`, the motif resembles:

Table 9: Information Content Matrix as relative divergence with a non-uniform background.

Position	1	2	3	4	5	6
A	1.030	0.366	0.000	0.680	0.680	0.366
C	0.000	0.000	0.541	0.000	0.000	0.000
G	0.000	0.000	1.028	0.000	0.000	0.000
T	0.000	0.000	0.090	0.000	0.000	0.000

Notice how the height of A in position 1 has decreased, and how the height of G in position 3 has increased. This makes sense when you consider that you are more likely to get an A and less likely to get a G to begin with.

6 Contribution weight matrices

A contribution weight matrix (CWM) stores signed, unnormalised contributions from an attribution analysis. Positive and negative entries represent effects in opposite directions under that analysis; they are neither base probabilities nor log-odds scores. Their interpretation depends on the model, prediction task, attribution method, and reference used to calculate them.

```
cwm.values <- matrix(c(1, -.5, 0, -.5, -.25, 1, -.5, 0), nrow = 4,
                     dimnames = list(c("A", "C", "G", "T"), NULL))
cwm <- create_motif(cwm.values, type = "CWM", name = "contributions")
view_motifs_lite(cwm, use.type = "CWM")
```

Specify `type = "CWM"` explicitly: signed matrices otherwise look like PWMs to the type detector. `read_matrix(type = "CWM")` and `read_meme(CWM = TRUE)` provide opt-in import, and `trim_cwm()` trims weak edge positions using absolute contributions. Site counts are unknown unless supplied; a column sum is not a number of contributing sequences.

`convert_type(cwm, "PPM")` normalises the **absolute** entries in each column (an all-zero column becomes uniform). This discards sign and absolute scale. It can provide a probability-matrix representation for downstream tools, but does not recover a binding model or make scan P-values probabilities of model attribution. There is no inverse conversion from a PPM to the original CWM. Retain the CWM alongside any derived PPM, preferably with `saveRDS()` when full precision and metadata matter.

References

Djordjevic, M., A. M. Sengupta, and B. I. Shraiman. 2003. "A Biophysical Approach to Transcription Factor Binding Site Discovery." *Genome Research* 13 (11): 2381–90.

Kullback, S., and R. A. Leibler. 1951. "On Information and Sufficiency." *The Annals of Mathematical*

Statistics 22: 79–86.

Ruan, S., and G. D. Stormo. 2017. "Inherent Limitations of Probabilistic Models for Protein-DNA Binding Specificity." *PLoS Computational Biology* 13 (7): e1005638.

Schneider, T. D. 1991. "Theory of Molecular Machines. II. Energy Dissipation from Molecular Machines." *Journal of Theoretical Biology* 148: 125–37.

Schneider, T. D., and R. M. Stephens. 1990. "Sequence Logos: A New Way to Display Consensus Sequences." *Nucleic Acids Research* 18: 6097–6100.

Schneider, Thomas D., Gary D. Stormo, Larry Gold, and Andrzej Ehrenfeucht. 1986. "Information Content of Binding Sites on Nucleotide Sequences." *Journal of Molecular Biology* 188: 415–31.

Shannon, Claude E. 1948. "A Mathematical Theory of Communication." *Bell System Technical Journal* 27: 379–423.

Stormo, G. D. 2000. "DNA Binding Sites: Representation and Discovery." *Bioinformatics* 16 (1): 16–23.

Stormo, Gary D., Thomas D. Schneider, Larry Gold, and Andrzej Ehrenfeucht. 1982. "Use of the Perceptron Algorithm to Distinguish Translational Initiation Sites in e. coli." *Nucleic Acids Research* 10: 2997–3011.

Touzet, H., and J.-S. Varre. 2007. "Efficient and Accurate p-Value Computation for Position Weight Matrices." *Algorithms for Molecular Biology* 2: 15.