RNA Bind-N-Seq (RBNS) Computational Pipeline

Enrichment (*R*) & library fraction tables; logo pipeline

Source code available at: https://bitbucket.org/pfreese/rbns_pipeline/src/master

FASTQ files of 20mers (or 40mers for some experiments) are separated into libraries for the input RNA pools and the pulled down sequences at each protein concentration based on the primer barcode hexamer introduced for PCR. For each protein concentration library, two measures of RBP binding are calculated for all kmers: 1. The "enrichment" or *R* value, and 2. The "estimated binding fraction". Each metric is calculated separately for all *k*s of interest (e.g., *k*=4, 5, 6, 7). A brief discussion of each, as well as the pipeline to create motif logos, is detailed below; see Lambert *et al.* Mol. Cell 2014 for a more in depth discussion (http://www.cell.com/molecular-cell/abstract/S1097-2765(14)00327-X).

Enrichment

To get the enrichment values of *k*mers for a library, the total number of occurrences of each *k*mer in all reads of the library is counted. Because there are 20-*k*+1 *k*mer instances per length 20 read and 4^k unique *k*mers, the sum of all counts over the 4^k *k*mers is (20-*k*+1)**X*, where *X* is the number of reads in the library. The counts are normalized to frequencies that sum to 1 within each library, and the enrichment of each $k\text{mer}_i$ is calculated as:

$$R = \frac{\text{frequency of } k\text{mer}_i \text{ in protein library}}{\text{frequency of } k\text{mer}_i \text{ in input library}}$$

Estimated Binding Fraction

The estimated binding fraction is calculated through Streaming *k*mer Analysis (SKA), an algorithm designed to estimate the fraction of RBNS reads which are bound at each of the 4^k possible *k*mers for a single RBNS library. The algorithm starts with a uniform distribution of binding weights over all *k*mers and iterates through the reads, incrementally adjusting the binding weights. The algorithm passes through the reads multiple times in order to converge on the correct distribution of weights. The first pass is used to obtain an initial estimate of the fraction of RBNS reads that are bound at each possible *k*mer, and subsequent passes are used to refine these until convergence.

The algorithm is as follows:

I. First Pass (*P* = 1):

1. The binding weights of all 4^k *k*mers are initialized with a pseudocount of 1.0. This is equivalent to assigning binding of one read to each *k*mer.

2. For each read *r*:

- All *k*mers present within the read's sequence are enumerated (one *k*mer beginning at each position from 1 to 20-*k*+1 in the length 20 read, denoted as the set $\{k\text{mers}\}_r$; generally each of these 20-*k*+1 *k*mers is unique, but sometimes two or more of the 20-*k*+1 *k*mers will be the same, in which case that *k*mer is counted proportional to its count in that read)

- The current weights of all those *k*mers are summed to obtain the normalization factor for read *r*, denoted as normalization_r

- For each *k*mer *j* in the set $\{k\text{mers}\}_r$, the weight of this kmer is updated from $\text{weight}_j \rightarrow \text{weight}_j * (1+1/\text{normalization}_r)$. In other words, the weight of 1 for read *r* is distributed to each of its constituent *k*mers proportional to those *k*mers' current weights, and these are then added to the current weights.

3. The weights of the 4^k *k*mers are normalized to sum to 1.

II. Second and subsequent passes (Pass *P*=2, 3, … until convergence):

1. A new set of weights for each of the 4^k *k*mers is initialized to 0.
2. For each read *r*:

- All *k*mers present within the read's sequence are enumerated as previously to obtain the set $\{k\text{mers}\}_r$

- The weights of all those *k*mers from the end of round *P*-1 are summed to obtain the normalization factor for read *r*, denoted as normalization_r

- For each *k*mer *j* in the set $\{k\text{mers}\}_r$, the weight of this kmer is updated from $\text{weight}_j \rightarrow \text{weight}_j (1+1/\text{normalization}_r)$.

3. The weights of the 4^k *k*mers are normalized to sum to 1.

The first vs. subsequent passes are identical except that the weight update upon occurrence of each *k*mer changes after *each read* in the first pass, whereas the increment update upon occurrence of each kmer changes only after the *entire pass* for subsequent passes. The latter update rule simply speeds up the computation for subsequent passes but does not alter convergence values as verified by simulations.

Generally, fewer than 10 passes are required for convergence, with quite good estimates after just 2 or 3 passes. The "estimated binding fraction" of a *k*mer is the normalized weight upon convergence, which roughly represents the fraction of library reads that were pulled-down due to protein binding to that *k*mer.

The intuition behind this algorithm is that more weight is attributed to *k*mers that have previously occurred more often (i.e., a "rich get richer" phenomenon), which is desirable since the *k*mers that are truly bound (e.g., UGCAUG for RBFox2) occur more frequently than do shifted "shadow" *k*mers that are enriched due to hitchhiking with the bound *k*mer but may not be sufficient on their own for binding (e.g., each of the four GCAUGN). In simulations, the iterative SKA algorithm suppresses the estimated binding fractions of these shifted *k*mers to near-background levels while the putative causal *k*mer estimated biding fractions are order(s) of magnitude higher.

Because the estimated binding fractions are calculated using the pulldown reads only without normalization to the input library, any nucleotide composition biases in the libraries will be present in the binding fractions and not normalized out as they are in the Enrichments described above. In particular, frequently the A nucleotide composition of the in vitro-transcribed RNA is about 5% higher than the 25% expected (~30%), with G composition correspondingly decreased by about 5% (~20%). Thus, **we recommend only using the estimated binding fractions for *k*mers with high enrichment and not for all *k*mers**, since polyA-containing *k*mers often have artificially high binding fractions due

to this nucleotide bias (particularly if the magnitude of the A-compositional bias is larger than that introduced from protein binding).

Motif Logos

Motif logos are made from aligning a set of significantly enriched *k*mers above a specified enrichment threshold cutoff and according to a set of rules as detailed below. These logos are independently created for multiple *k* lengths and for *k*mers enriched above various Z-score thresholds, where the Z-score is derived from the mean & standard deviation of the *R* values over all 4^k kmers (typically run for at least *k*=5 and 6 with Z-score = 2 and 3). For simplicity, only the logo made from parameters: *k* = 5mer, enrichment Z-score = 3 is uploaded to the ENCODE DCC website as a "Document" (based on visual inspection of dozens of RBNS experiments, this *k* and Z-score threshold appears to best at balancing the competing interests of capturing the diversity of the most highly enriched *k*mers that the RBP should be able to bind yet not being too lax to include more lowly-enriched *k*mers of different composition that are likely false positives). Below is a description of the logo pipeline in words; a more visual description can be found on the final page of this document.

For a particular *k* and Z-score, the logo is made from aligning a set of enriched *k*mers as determined through a stepwise procedure of: masking the most enriched *k*mer & recalculating enrichments, continuing until the highest enrichment falls below a pre-specified threshold.

Enrichments for all *k*mers are calculated (for the pulldown library RBP concentration that has the highest overall enrichment), including the mean and standard deviation of all 4^k *k*mers to determine the enrichment at the desired Z-score cutoff (cutoff = mean + Z*st.dev.). The top enriched *k*mer and its enrichment above background (= *R*-1, since an *R* value of 1 means equal frequency in pulldown & input) as its weight is used as the founding *k*mer for the first logo. To avoid including "shadow" *k*mers that are shifted versions of *k*mers already accounted for (e.g., GGGGU offset by 1 from GGGGG), all instances of the most enriched *k*mer are X'ed out in the pulldown and input libraries. Enrichments are then recalculated on the modified pulldown & input libraries, the top *k*mer and its recalculated enrichment above background are removed, X'ed out, etc., continuing until the enrichment of the top remaining *k*mer is less than the Z-score cutoff previously determined.

This set of *k*mers is then aligned sequentially, beginning with the most enriched *k*mer as the "founding *k*mer" of the 1st logo. An attempt is made to align the next most enriched *k*mer to the founding *k*mer of the 1st logo according to the following rules (these rules are for 5mers; rules for aligning 6mers are detailed further below):

1. Offset of 1; 0 mismatches to founding *k*mer
2. Offset of 0; 1 mismatch
3. Offset of 2; 0 mismatches among 3 aligned positions
4. Offset of 1; 1 mismatch among 4 aligned positions

If the second *k*mer can be aligned to the first according to one of the above rules, it is; if not, a 2nd logo is started with that as the "founding *k*mer". This process is sequentially repeated for all of the enriched *k*mers in the order that they were X'ed out above, attempting to align that *k*mer to one of the existing logos, or starting a new logo if not possible.

For each logo, which is made up of a set of aligned *k*mers each with a corresponding enrichment above background, a Position Weight Matrix (PWM) is made by weighting the aligned nucleotides at each position by its enrichment above background (25% of each nucleotide goes toward the PWM if there's a leading or lagging offset for the *k*mer at that position). Any leading or lagging positions that don't have weighted alignments of at least 25% (i.e., if more than 75% is padded) are removed.

To increase the robustness of the logos generated, the pulldown & input reads are each divided into halves, the procedure is carried out on both halves independently, and only *k*mers that appear in both logos are included in the final logo (the enrichment above background values as weights for each *k*mer are averaged over the two halves).

As an example (also see schematic on last page), for the first half of the FUBP3 80 nM pulldown & input libraries, 5mers have an original mean R=0.91 & standard deviation=0.6, so the enrichment Z-score=3 threshold is R=2.71.

The stepwise enrichments (X'ing out each enriched *k*mer before recalculating and taking the most enriched remaining *k*mer) until reaching the R=2.71 threshold are:

UAUAU	5.056
UUUUU	4.144
UUUAU	3.838
AUUAU	3.390
UAAUA	3.399
UAAAU	3.279
UUAAU	2.979
UAUGU	2.978
UAUAA	2.893
UAUUU	2.888

Subtracting 1 from each of the enrichments to get the enrichments above background and aligning them according the above rules yields:

Logo 1

__UAUAU	4.056
AUUAU__	2.390
UAAUA	2.399
__UAAAU	2.279
__UAUGU	1.978
__UAUAA	1.893

Logo 2

UUUUU__	3.144
UUUAU__	2.838
__UAUUU	1.888

Logo 3
UUAAU 1.979

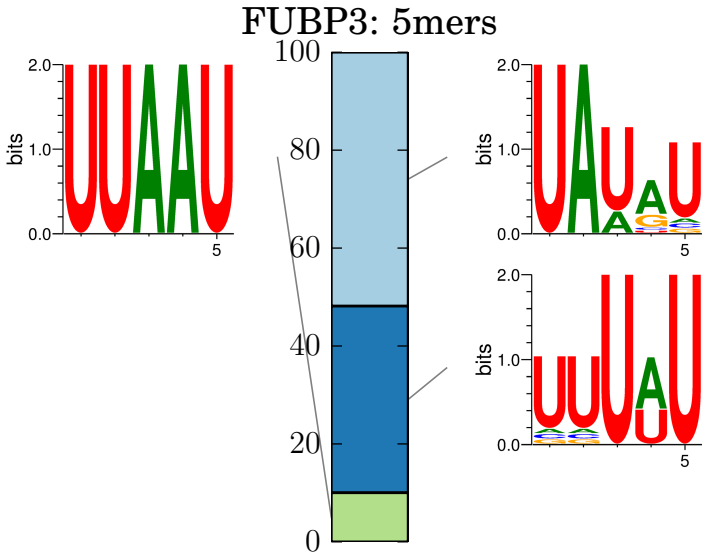
The same procedure is carried out independently on the second half of the pulldown & input reads. The *k*mers that occur in both halves are used to make a composite alignment (with enrichment above background values averaged), in this case yielding:

Logo 1
__UAUAU 4.053
AUUAU__ 2.375
__UAAAU 2.216
__UAUGU 1.976

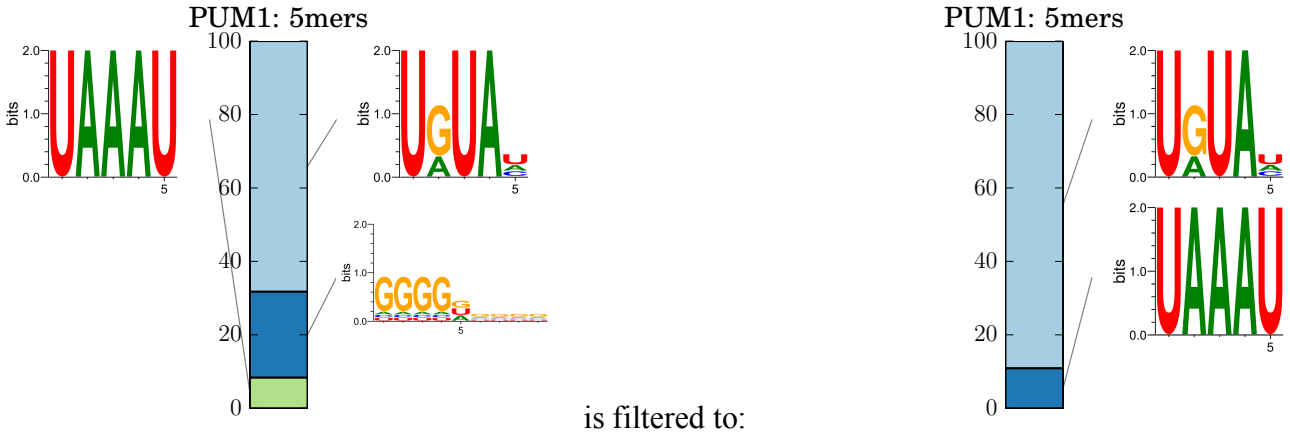
Logo 2
UUUUU__ 3.125
UUUAU__ 2.821
__UAUUU 1.861

Logo 3
UUAAU 2.051

Each logo's fraction in the bar graph is proportional to the sum of its constituent *k*mer *R*-1 values (normalized 10.62 vs. 7.81 vs. 2.051):



Note: occasionally, poly(X) *k*mers will be enriched just above the enrichment threshold despite having very different composition from the top enriched motifs and being quite far down in the original enrichment tables (e.g., #50). RBPs that have poly(X) and related *k*mers as most enriched generally have much higher enrichments (e.g. HNRNPH2 has top *k*mer GGGGG R=8). In contrast, cases in which a poly(X) *k*mer occurs just above the enrichment threshold (e.g., GGGGG after a number of A/U-rich *k*mers) generally have much lower enrichments for the poly(X) *k*mer (e.g., R = 1.5) and also have different enrichment profiles over the 5 RBP concentrations (5/20/80/320/1300 nM [RBP]) than the other, non-poly(X) significantly enriched *k*mers. These are therefore assumed to be an artifact and manually removed from the logos, e.g. PUM1:



Rules for aligning 6mers:

1. Offset of 1; 0 mismatches to founding *k*mer among 5 aligned positions
2. Offset of 0; 1 mismatch
3. Offset of 2; 0 mismatches among 4 aligned positions
4. Offset of 1; 1 mismatch among 5 aligned positions
5. Offset of 0; 2 mismatches
6. Offset of 2; 1 mismatch among 4 aligned positions

RBNS Logo Pipeline

Logos dispalyed are typically for parameters *k* = 5 and Z-score threshold = 3.0

Example for FUBP3:

Divide pulldown & input each into 2 halves to filter for robust significantly enriched *k*mers

- Run independently on each half
- Only *k*mers in both halves included in final logos (see below)

Original Enrichments (*R*) from most enriched Pulldown library

UAUAU	5.056
UUUAU	4.246
AUAUA	4.201
UUUUU	4.080
UAUUU	3.994
...	
CGCCG	0.416
CGUCG	0.386

mean[R] = 0.91
St. Dev.[R] = 0.60

Z-score 3
R Threshold: *2.71*

Original Enrichments (*R*) from most enriched Pulldown library

UAUAU	5.056
UUUAU	4.246
AUAUA	4.201
UUUUU	4.080
UAUUU	3.994
...	
CGCCG	0.416
CGUCG	0.386

Above *R* Threshold: *2.71* ✓

X out top *k*mer in pulldown & input reads

Pulldown reads	Input reads
UGGGCGGCGCUCUCACUCGA	ACUGUGUAUAAACAUCUAUA
UAUAUUUGUACUAUGGAGUG	GUACCAUUGAAUAACACAAG
GAACGAAAAUGCUAAAUGCG	GAUCAGCGAAGAACUAGCAA
CUAAUUAUAAAGUGAAGAUC	CGUGGCUGAGUAUGGAGCCC
UAAAUGCUAGCCGGUUCAAA	AUACCGCAAAGGCAAGAAUG
AGCAUUAA**UAUAU**AGUUAUA	UCAUCUUCACGCCGAAAUCG
AUCCCCGUUACUUAUUAUUU	UAUAAUGAAGUCAG**UAUAU**U
UACACAAACGUCAAACCCGG	UUAGCAAGAGUUACGCCACG
CUCACAAUUAAAGUCCCCCA	ACCAAAAUAUUGAGUGCCUA
AGAACCCUA**UAUAU**AAGAUA	CUUGCAAGCGUCCCGUGGCA

Pulldown reads	Input reads
UGGGCGGCGCUCUCACUCGA	ACUGUGUAUAAACAUCUAUA
XXXXXUUGUACUAUGGAGUG	GUACCAUUGAAUAACACAAG
GAACGAAAAUGCUAAAUGCG	GAUCAGCGAAGAACUAGCAA
CUAAUUAUAAAGUGAAGAUC	CGUGGCUGAGUAUGGAGCCC
UAAAUGCUAGCCGGUUCAAA	AUACCGCAAAGGCAAGAAUG
AGCAUUAA**XXXXX**AGUUAUA	UCAUCUUCACGCCGAAAUCG
AUCCCCGUUACUUAUUAUUU	UAUAAUGAAGUCAG**XXXXX**U
UACACAAACGUCAAACCCGG	UUAGCAAGAGUUACGCCACG
CUCACAAUUAAAGUCCCCCA	ACCAAAAUAUUGAGUGCCUA
AGAACCCUA**XXXXX**AAGAUA	CUUGCAAGCGUCCCGUGGCA

Recalculate enrichments

UUUUU	4.14
UUUAU	4.05
UAUUU	3.77
...	
CGCCG	0.43
CGUCG	0.40

Above *R* Threshold: *2.71* ✓

Continue X'ing out most enriched *k*mer and recalculating enrichments

...

Recalculate enrichments

UUAAA	2.65
UUAAA	2.54
UAAAA	2.51
...	
CGCCG	0.45
CGUCG	0.42

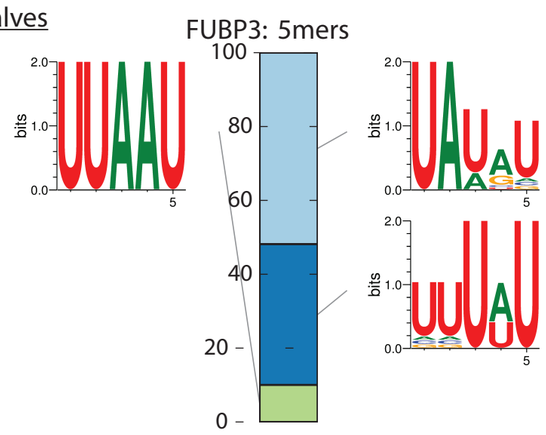
Above *R* Threshold: *2.71* ✗ END and align *k*mers

Half 1		Half 2	
Logo 1	***R* - 1**	**Logo 1**	***R* - 1**
__UAUAU	4.056	__UAUAU	4.049
AUUAU__	2.390	__AAUAU	2.393
UAAUA	2.399	AUUAU__	2.360
__UAAAU	2.279	__UAAAU	2.154
__UAUGU	1.978	__UAUGU	1.973
__UAUAA	1.893		
Logo 2		**Logo 2**	
UUUUU__	3.144	UUUUU__	3.105
UUUAU__	2.838	UUUAU__	2.804
__UAUUU	1.888	__UAUUU	1.835
Logo 3		**Logo 3**	
UUAAU	1.979	UUAAU	2.124
		AUAAU	1.926

Filter for *k*mers in both halves

Logo 1	***R* - 1**
__UAUAU	4.053
AUUAU__	2.375
__UAAAU	2.216
__UAUGU	1.976
Logo 2	
UUUUU__	3.125
UUUAU__	2.821
__UAUUU	1.861
Logo 3	
UUAAU	2.051

FUBP3: 5mers



Sequentially align 5mers (or start new logo) according to:

Align 5mer to founding *k*mer of existing motif (DEFGH) if:

0 offset	1 offset		2 offsets	
DEFGH	DEFGH	DEFGH	DEFGH	DEFGH
DEFGH	CDEFG-	-EFGHI	BCDEF--	--FGHIJ

if no alignment possible, use 5mer as founder of new logo

- 0 offset: 1 mismatch allowed in 5 aligned positions (i.e., one of D/E/F/G/H can be different than D/E/F/G/H)
- 1 offset: At most 1 mismatch allowed in 4 aligned positions (i.e., one of D/E/F/G or E/F/G/H, can differ from D/E/F/G or E/F/G/H, respectively)
- 2 offsets: 0 mismatches allowed in 3 aligned positions (i.e., DEF or FGH must all match DEF or FGH, respectively)

Create PWM for each sequence logo using excess enrichments (*R*-1) as weights. Offset positions (i.e., "_" in alignments) contribute 25% of each nt toward alignment. Lagging or leading positions having more than 75% offset weight are removed (e.g., the leading "AU" in Logo 1 has only 2.375/10.62 = 22% specified; likewise for the lagging "UU" in Logo 2). Each logo's fraction in the bar graph is proportional to the sum of its constituent *k*mer *R*-1 values (normalized 10.62 vs. 7.81 vs. 2.051).