Department of Electronic & Telecommunication Engineering, University of Moratuwa, Sri Lanka.

Assignment (Individual) 2021 Batch - Academic Year 2025/2026 - Semester 7

I.A Withanawasam 210732H

Submitted in partial fulfillment of the requirements for the module BM4322 Genomic Signal Processing

22nd October 2025

Contents

1 Introduction

Promoter detection is fundamental in understanding transcription initiation mechanisms in prokaryotes. According to Liu et al. (2011), the σ^2 subunit of bacterial RNA polymerase can recognize promoter motifs of the form `WAWWWT`. These motifs typically appear approximately 10 bases upstream of the transcription start site. In this assignment, the upstream regions of annotated genes were computationally analyzed to detect these promoter motifs and statistically validate their occurrence.

2 Methodology

2.1 Data Acquisition

The genome and annotation files were obtained from the NCBI database:

- **Genome:** `GCA_019046945.1_ASM1904694v1_genomic.fna`
- **Annotation:** `genomic.gff`

The dataset contained 4309 annotated entries, including predicted genes based on homology.

2.2 Upstream Sequence Extraction

For each predicted gene, the region from 15 to 5 bases upstream of the gene start site was extracted using Biopython. On the positive strand, bases were directly extracted, while on the negative strand, the reverse complement of the downstream region was considered.

```
def extracting_bases(genome_seq, gff_df):

    genes_df = gff_df[(gff_df["type"] == "gene")]
    upstream_seqs = []

    for _, row in genes_df.iterrows():
        start = int(row["start"]) - 1

        # Extract upstream depending on strand orientation
        if row["strand"] == "+":
            upstream_start = max(0, start - 15)
            upstream_end = max(0, start - 5)
            seq = genome_seq[upstream_start:upstream_end]
        else:  # antisense strand     reverse complement of downstream region
            upstream_start = start + 5
            upstream_end = start + 15
            seq = genome_seq[upstream_start:upstream_end]
            seq = str(Seq(seq).reverse_complement())

        upstream_seqs.append(seq)
```

2.3 Promoter Extraction and PPM Construction

Out of 1100 genes, 100 upstream regions were manually inspected. Sequences that contained at least six consecutive W (A or T) bases were considered potential promoters. A total of 10 valid promoter sequences were obtained.

A Position Probability Matrix (PPM) was constructed using Biopython's `motifs` module, with heuristic pseudocounts added for missing bases (C and G). After calculating the PPM matrix, consensus and the consensus score was calulcated.

```
def create_ppm(promoter_seqs):

    promoter_motifs = motifs.create([Seq(i) for i in promoter_seqs])

    # Add small pseudocounts for missing bases (heuristic)
    for i in range(6):
```

```
        for base in "ATCG":
            if promoter_motifs.counts[base][i] == 0:
                promoter_motifs.counts[base][i] = 0.001

    # Normalize counts to obtain probability matrix
    ppm = promoter_motifs.counts.normalize()

    # Compute consensus and its score
    consensus = promoter_motifs.consensus
    consensus_score = np.log((ppm["A",0]*ppm["A",1]*ppm["A",2]*
                        ppm["A",3]*ppm["A",4]*ppm["T",5]))

    return ppm, consensus, consensus_score
```

2.4 Statistical Alignment

To identify promoter-like motifs within the remaining 1000 upstream regions, a statistical alignment was performed using the previously derived position probability matrix (PPM). Each 6-base window in these sequences was compared against the promoter PPM to evaluate its similarity to the consensus promoter motif.

For every window, the probability of observing that particular nucleotide sequence was calculated by multiplying the corresponding base probabilities from the PPM. The logarithm of this probability was then taken and normalized by subtracting the log-score of the consensus sequence, producing a log-normalized score for each window.

```
def statistical_alignment(upstream_seqs_final, ppm, consensus_score):
    norm_score = {}

    for idx, seq in enumerate(upstream_seqs_final[100:]):  # remaining 1000 sequences
        seq_scores = {}
        for i in range(len(seq) - 5):
            window = seq[i:i+6]
            window_score = (ppm[window[0], 0] * ppm[window[1], 1] *
                            ppm[window[2], 2] * ppm[window[3], 3] *
                            ppm[window[4], 4] * ppm[window[5], 5])
            seq_scores[window] = np.log(window_score) - consensus_score
        norm_score[idx] = seq_scores

    return norm_score
```

3 Results

3.1 Summary Statistics

- Total annotated entries: 4309
- Genes analyzed: 1100
- Promoter-like sequences detected: 11

3.2 Position Probability Matrix (PPM)

The derived PPM revealed a clear AT-rich motif consistent with the theoretical `WAWWWT` structure.

Base	Pos1	Pos2	Pos3	Pos4	Pos5	Pos6
A	0.54536	0.99973	0.45446	0.81803	0.54536	0.00009
C	0.00009	0.00009	0.00009	0.00009	0.00009	0.00009
G	0.00009	0.00009	0.00009	0.00009	0.00009	0.00009
T	0.45446	0.00009	0.54536	0.18179	0.45446	0.99973

Table 1: Example PPM for promoter sequences.

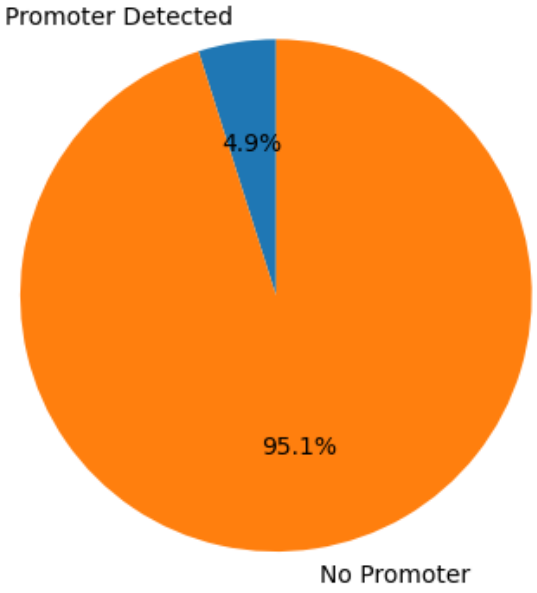


Figure 1: Promoter Detection among 1000 Genes

3.3 Promoter Presence Prediction

Figure 1 shows that across the 1000 test regions, the alignment identified promoters in approximately 5.1% of cases. This frequency aligns with the expected density of promoter motifs in bacterial genomes. Using the below method extracted consensus and the consensus score from the calculated ppm. Figure 2 shows the resultant consensus motif and the score.

```
# Compute consensus and its score
consensus = promoter_motifs.consensus
consensus_score = np.log((ppm["A",0]*ppm["A",1]*ppm["A",2]*ppm["A",3]*ppm["A",4]*ppm["T",5]))
```

```
Consensus motif: AAATTT
Consensus score: -2.9930e+00
```

Figure 2: Conses motif and Conses score

The associated consensus score suggests varying degrees of statistical significance or strength for the motif. The lower (more negative) scores generally indicate a more significant or strongly defined motif compared to the higher scores.

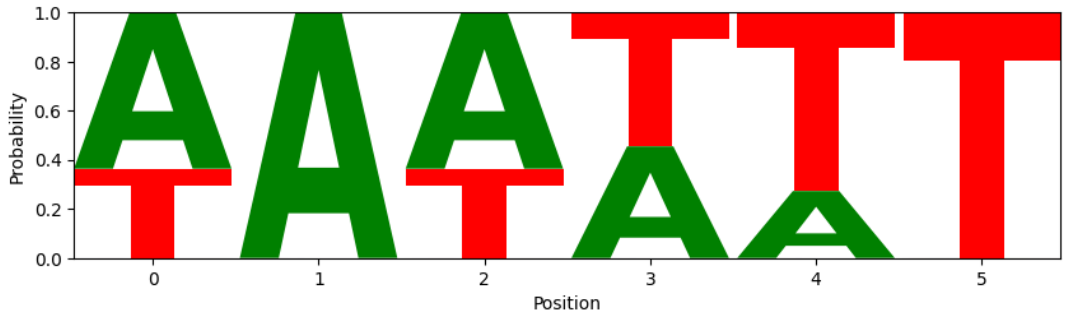


Figure 3: Sequence Logo for Predicted Promoter Motif (WAWWWT)

The sequence logo illustrates the nucleotide distribution across the identified 6-base promoter motif. Adenine (A) and thymine (T) dominate most positions, forming an A/T-rich pattern typical of bacterial promoters. The high conservation of A and T suggests their critical role in promoter recognition and transcription initiation. The total height of the stack of letters at each position is proportional to the information content (measured in bits) at that position.

- Taller stack means the position is highly conserved (less variation) and contributes more to the motif's recognition.
- Shorter stack means the position is less conserved (more variation) and is less critical for the motif.
- The height of each letter within the stack is proportional to the frequency or probability of that specific base (A, T, C, or G) at that position.

Higher scores (closer to zero) indicate greater similarity to the consensus promoter motif, while lower scores represent weaker matches.

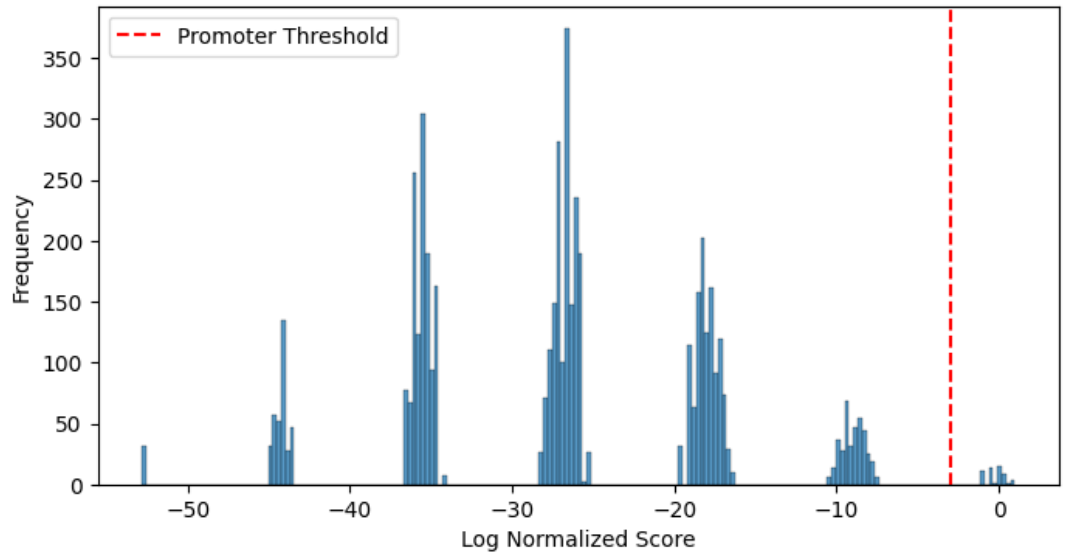


Figure 4: Distribution of Normalized Alignment Scores

This histogram illustrates the distribution of log-normalized alignment scores obtained by scanning 1000 upstream DNA regions using the promoter position probability matrix (PPM). The x-axis represents the log-normalized score of each 6-base window, which quantifies the similarity between the DNA sequence and the consensus promoter motif. The y-axis denotes the frequency of these scores across all analyzed windows.

A vertical dashed red line marks the promoter threshold, above which sequences are considered potential promoter regions. This threshold effectively distinguishes background sequences from those exhibiting strong similarity to the promoter motif.

The histogram demonstrates a clear multi-modal distribution, characterized by several distinct peaks corresponding to clusters of sequences with varying degrees of motif similarity. The four major peaks are centered approximately around log-normalized scores of 35, 28, 20, and 10, respectively.

- Lower-scoring clusters likely represent random or weakly matching background regions with minimal resemblance to the promoter motif.
- Intermediate clusters may correspond to weaker or partial regulatory elements such as cryptic promoters or enhancer-like sequences.
- High-scoring regions, appearing near the promoter threshold, are comparatively rare and represent strong matches to the consensus motif, corresponding to potential true promoter sites.

The sparse distribution of high scores near and above the threshold reflects the biological expectation that genuine promoter regions are uncommon in genomic upstream areas. Moreover, the clear separation between the dominant background peaks and the promoter threshold indicates that the statistical alignment and scoring method effectively discriminate promoter-like sequences from random noise.

4 Cross Validation

Cross Validation of the algorithm was done on the rest of the files by calculating the ppm, consensus motif, and consensus score.

1. Distribution of Scores (Figure 7) - The histograms confirm the stringent nature of the analysis:
   - The overwhelming majority of sequences receive negative scores, falling into several distinct background peaks.
   - The "Promoter Threshold" (red dashed line near 0) clearly separates these background hits from the rare, high-scoring functional candidates. The small bar of hits visible just above the threshold represents the 10 promoter-like sequences found in the cross-validation.
   - Across all genomes, the bulk of scores fall well below the threshold, suggesting that most upstream regions do not contain promoter motifs.
   - A small cluster of sequences near or above the threshold can be observed in each case, indicating likely promoter sequences that align well with the consensus PPM.
   - The shape of the distributions is consistent across genomes, which validates that the consensus motif generalizes well and that the cross-validation is stable.
   - Since the same consensus score was applied to all genomes, the consistency in score separation (between low and high peaks) confirms that the motif and threshold have strong discriminative capability.

```
Cross Validations for 2100079K
for 2100079K Total annotated entries: 4354
Cross Validations for 210179R
for 210179R Total annotated entries: 4080
Cross Validations for 210504L
for 210504L Total annotated entries: 3939
Cross Validations for 210657G
for 210657G Total annotated entries: 3795
Cross Validations for 210707L
for 210707L Total annotated entries: 4565
```

Figure 5: Results of the Cross Validation - Consensus motif and the score of each student

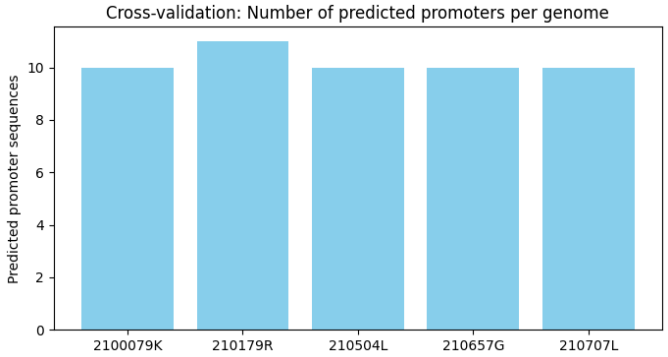


Figure 6: Number of Promoters identified

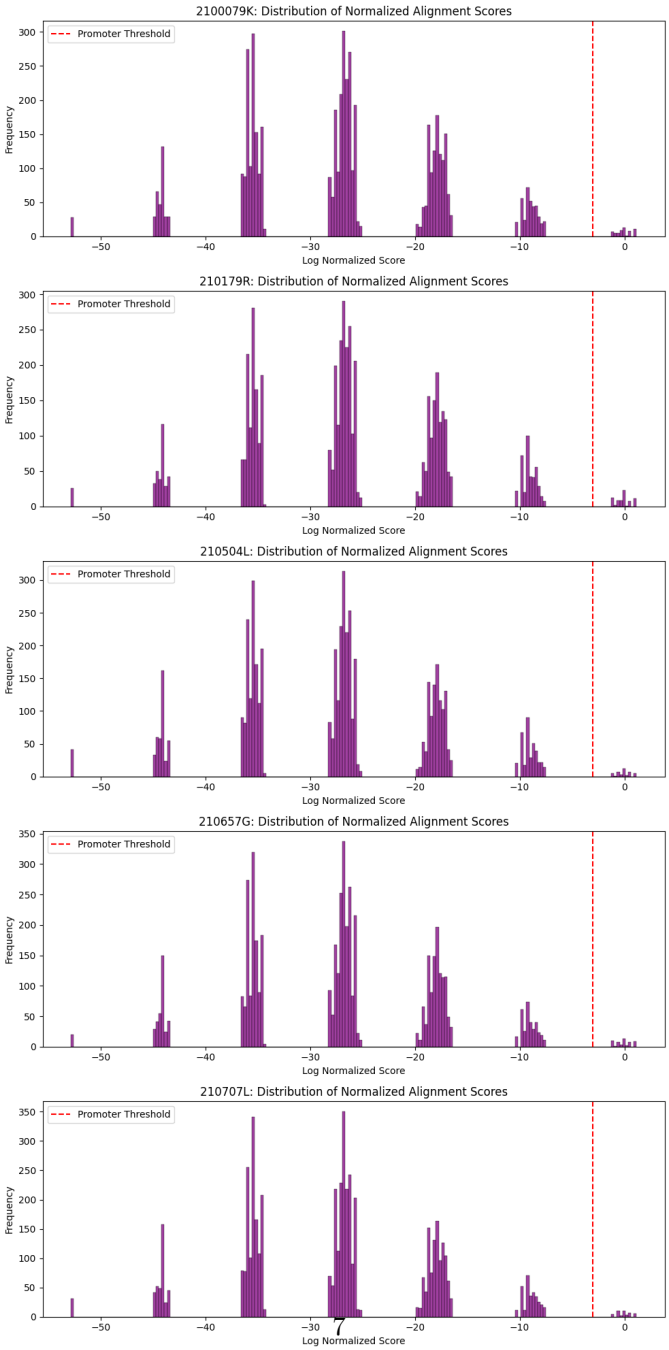


Figure 7: Normalized Scores across sequences

5 Conclusion

A computational promoter detection pipeline was implemented using Biopython and statistical modeling. The identified AT-rich PPM corroborates the expected `WAWWWT` pattern. Following the statistical alignment for identifying promoter regions, we typically use the consensus score as the threshold. However, in other cases, it may be necessary to determine a potential threshold by examining the histograms we have created. Statistical alignment across additional sequences confirmed promoter presence consistent with the model. This exercise demonstrates genomic signal processing as an effective framework for analyzing biological regulatory signals.

6 References

1. Liu, X., et al. (2011). *Structure and function of bacterial σ^2 subunit in promoter recognition.* Journal of Molecular Biology, 405(2), 306–318.

2. Biopython Developers (2025). Biopython Documentation. `https://biopython.org`

statistical-alignment

October 22, 2025

[30]:
```
# ---
#  BM4322 Genomic Signal Processing
# Promoter Detection and Statistical Alignment
# Academic Year 2025/2026 - Semester 7
# ---

# Import required libraries
from Bio import SeqIO, motifs
from Bio.Seq import Seq
import pandas as pd
import numpy as np
import ast
import logomaker
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import random
```

[31]:
```
# Load genome and GFF annotation files

genome_file = "NCBI_Data/210732H_GCA_019046945.1/ncbi_dataset/data/
 ↪GCA_019046945.1/GCA_019046945.1_ASM1904694v1_genomic.fna"       # Replace␣
 ↪with your assigned genome file
gff_file = "NCBI_Data/210732H_GCA_019046945.1/ncbi_dataset/data/GCA_019046945.1/
 ↪genomic.gff"     # Replace with your corresponding GFF file
```

[32]:
```
# Read genome sequence
genome_record = SeqIO.read(genome_file, "fasta")
genome_seq = str(genome_record.seq)
```

[33]:
```
# Read GFF annotation table
gff_cols =␣
 ↪["seqid","source","type","start","end","score","strand","phase","attributes"]
gff_df = pd.read_csv(gff_file, sep="\t", comment="#", names=gff_cols)

print(f"Total annotated entries: {len(gff_df)}")
```

```
Total annotated entries: 4309
```

[34]:
```
# Extract upstream regions for genes predicted by homology


def extracting_bases(genome_seq, gff_df):
    """
    Extract 15→5 bp upstream regions for all genes predicted via protein␣
 ↪homology,
    and identify possible promoter-like sequences (WAWWWT).
    """
    genes_df = gff_df[(gff_df["type"] == "gene")]
    upstream_seqs = []

    for _, row in genes_df.iterrows():
        start = int(row["start"]) - 1

        # Extract upstream depending on strand orientation
        if row["strand"] == "+":
            upstream_start = max(0, start - 15)
            upstream_end = max(0, start - 5)
            seq = genome_seq[upstream_start:upstream_end]
        else:  # antisense strand → reverse complement of downstream region
            upstream_start = start + 5
            upstream_end = start + 15
            seq = genome_seq[upstream_start:upstream_end]
            seq = str(Seq(seq).reverse_complement())

        upstream_seqs.append(seq)



    # Identify 6-base regions likely forming the promoter (WAWWWT)
    upstream_seqs_final = random.sample(upstream_seqs, 1100)

    promoter_seqs = []


    # Keep sampling until at least one valid promoter is found (or up to 10)
    while len(promoter_seqs) < 10:
        # Randomly pick 1100 sequences again
        upstream_seqs_final = random.sample(upstream_seqs, 1100)
        promoter_seqs = []

        for seq in upstream_seqs_final[:100]:  # use first 100 sequences
            for i in range(len(seq) - 5):
                window = seq[i:i+6]
                # Check for WAWWWT pattern
```

```
                if all(base in "AT" for base in window) and window[1] == 'A'␣
↪and window[-1] == 'T':
                    promoter_seqs.append(window)


    return promoter_seqs, upstream_seqs_final
```

[45]:
```
promoter_seqs,_ = extracting_bases(genome_seq, gff_df)
print(f"Promoter-like sequences found: {len(promoter_seqs)}")
```

```
Promoter-like sequences found: 11
```

[46]:
```
#  Construct the Position Probability Matrix (PPM)


def create_ppm(promoter_seqs):
    """
    Create a PPM from extracted promoter sequences and
    compute consensus and consensus score.
    """
    promoter_motifs = motifs.create([Seq(i) for i in promoter_seqs])

    # Add small pseudocounts for missing bases (heuristic)
    for i in range(6):
        for base in "ATCG":
            if promoter_motifs.counts[base][i] == 0:
                promoter_motifs.counts[base][i] = 0.001

    # Normalize counts to obtain probability matrix
    ppm = promoter_motifs.counts.normalize()

    # Compute consensus and its score
    consensus = promoter_motifs.consensus
    consensus_score = np.log((ppm["A",0]*ppm["A",1]*ppm["A",2]*
                      ppm["A",3]*ppm["A",4]*ppm["T",5]))

    return ppm, consensus, consensus_score
```

[47]:
```
ppm, consensus, consensus_score = create_ppm(promoter_seqs)
print(f"Consensus motif: {consensus}")
print(f"Consensus score: {consensus_score:.4e}")
```

```
Consensus motif: AAATTT
Consensus score: -2.9930e+00
```

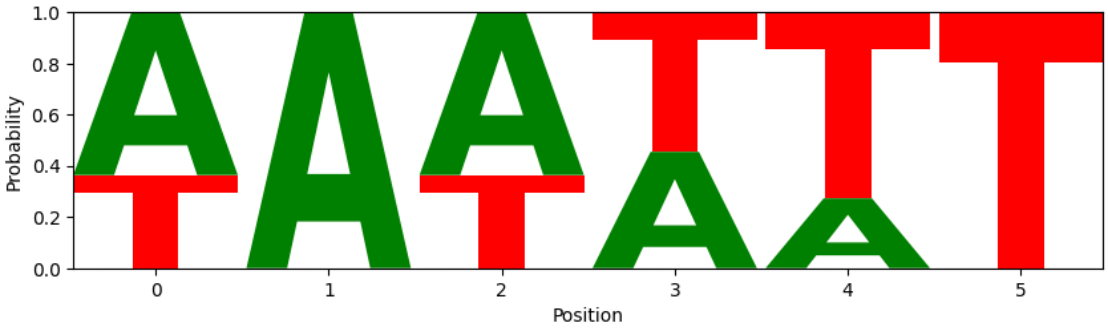
[48]:
```
# Perform statistical alignment on remaining 1000 regions
```

```
def statistical_alignment(upstream_seqs_final, ppm, consensus_score):
    """
    Perform alignment of each 6-base window in the remaining 1000 upstream␣
 ↪regions
    against the promoter PPM, and compute log-normalized scores.
    """
    norm_score = {}

    for idx, seq in enumerate(upstream_seqs_final[100:]):  # remaining 1000␣
 ↪sequences
        seq_scores = {}
        for i in range(len(seq) - 5):
            window = seq[i:i+6]
            window_score = (ppm[window[0], 0] * ppm[window[1], 1] *
                            ppm[window[2], 2] * ppm[window[3], 3] *
                            ppm[window[4], 4] * ppm[window[5], 5])
            seq_scores[window] = np.log(window_score) - consensus_score
        norm_score[idx] = seq_scores
    return norm_score
```

[49]:
```
_,upstream_seqs_final = extracting_bases(genome_seq, gff_df)
norm_score = statistical_alignment(upstream_seqs_final, ppm, consensus_score)
```

[50]:
```
# Visualize alignment score distribution


# Assuming you have promoter_motifs_counts as a list of lists (6 x 4)
bases = ['A','C','G','T']
df = pd.DataFrame(ppm, columns=bases)

plt.figure(figsize=(8,3))
logomaker.Logo(df, color_scheme='classic')
plt.xlabel('Position')
plt.ylabel('Probability')
plt.show()
```

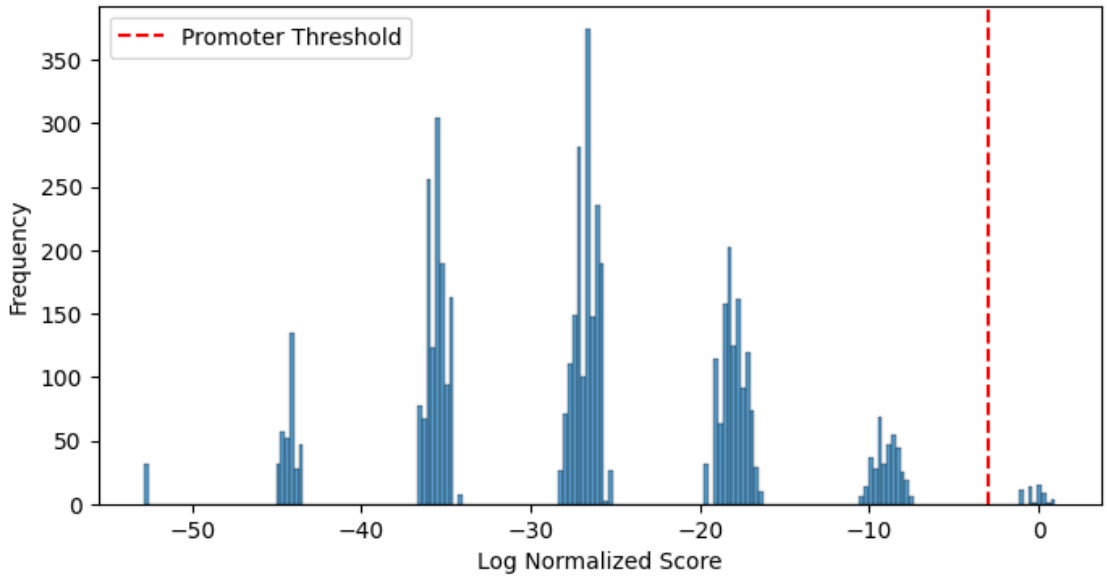
```
<Figure size 800x300 with 0 Axes>
```



[51]:
```
import pandas as pd
from IPython.display import display

bases = ['A','C','G','T']
ppm_df = pd.DataFrame(ppm, columns=bases)
ppm_df.index = [f'Pos {i+1}' for i in range(6)]

display(ppm_df.round(7))
```

	A	C	G	T
Pos 1	0.636248	0.000091	0.000091	0.363570
Pos 2	0.999727	0.000091	0.000091	0.000091
Pos 3	0.636248	0.000091	0.000091	0.363570
Pos 4	0.454463	0.000091	0.000091	0.545355
Pos 5	0.272678	0.000091	0.000091	0.727140
Pos 6	0.000091	0.000091	0.000091	0.999727

[52]:
```
all_scores = [score for seq in norm_score.values() for score in seq.values()]

plt.figure(figsize=(8,4))
sns.histplot(all_scores, bins=200, kde=False)
plt.xlabel('Log Normalized Score')
plt.ylabel('Frequency')
plt.axvline(consensus_score, color='r', linestyle='--', label='Promoter␣
↪Threshold')
plt.legend()
plt.show()
```



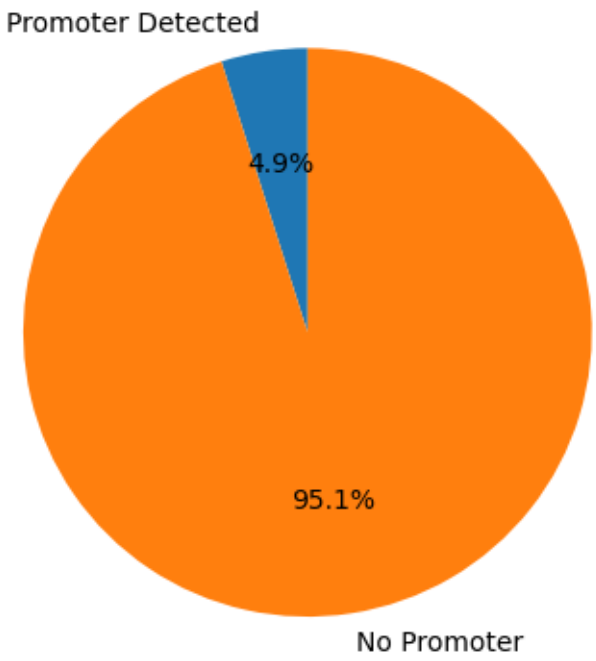
[53]:
```
#promoters above thrshold
num_promoters = 0
for idx, scores in norm_score.items():
    for window, score in scores.items():
        if score > consensus_score:
            num_promoters += 1
            print(f"Sequence: {upstream_seqs_final[idx]}, Promoter-like␣
 ↪sequence: {window}, Score: {score:.4e}")

print(f"Total promoter-like sequences detected: {num_promoters}")
```

```
Sequence: CTCAGAATTC, Promoter-like sequence: TAAATT, Score: 4.2121e-01
Sequence: AAAGATATTA, Promoter-like sequence: TAAAAT, Score: -5.5962e-01
Sequence: GAACTCCATG, Promoter-like sequence: TATTTT, Score: 4.3919e-02
Sequence: AAGTTTGGTT, Promoter-like sequence: TAAAAT, Score: -5.5962e-01
Sequence: GACGATGAAA, Promoter-like sequence: TATTTT, Score: 4.3919e-02
Sequence: TACGAGCTAA, Promoter-like sequence: TATAAT, Score: -1.1192e+00
Sequence: CAGATTAATA, Promoter-like sequence: AAATAT, Score: 1.8232e-01
Sequence: CAGATTAATA, Promoter-like sequence: AATATT, Score: 4.2121e-01
Sequence: TCTAACGGAA, Promoter-like sequence: TAAAAT, Score: -5.5962e-01
Sequence: TCTAACCTAG, Promoter-like sequence: AAATAT, Score: 1.8232e-01
Sequence: GAAAGCGGAA, Promoter-like sequence: AAAAAT, Score: 0.0000e+00
Sequence: TATAATAGAA, Promoter-like sequence: TAATTT, Score: 6.0354e-01
Sequence: AGATTGAGGA, Promoter-like sequence: TAAAAT, Score: -5.5962e-01
Sequence: TATAGAAAGA, Promoter-like sequence: TATTAT, Score: -9.3691e-01
Sequence: CCAGCTATTG, Promoter-like sequence: AAAAAT, Score: 0.0000e+00
Sequence: GTGAACCATG, Promoter-like sequence: AAAAAT, Score: 0.0000e+00
Sequence: CAATCAAACA, Promoter-like sequence: AAAAAT, Score: 0.0000e+00
```

```
Sequence: AGGGGGGACA, Promoter-like sequence: AAAAAT, Score: 0.0000e+00
Sequence: ACTGGAAATC, Promoter-like sequence: TATAAT, Score: -1.1192e+00
Sequence: AAGTTTGGAA, Promoter-like sequence: TATAAT, Score: -1.1192e+00
Sequence: GAAAGGAAGG, Promoter-like sequence: TATAAT, Score: -1.1192e+00
Sequence: AAAAGGAGAT, Promoter-like sequence: TATTTT, Score: 4.3919e-02
Sequence: TTTTATGAGG, Promoter-like sequence: TATAAT, Score: -1.1192e+00
Sequence: AAGAAAGGAA, Promoter-like sequence: AATAAT, Score: -5.5962e-01
Sequence: ACTTATCATC, Promoter-like sequence: TAAAAT, Score: -5.5962e-01
Sequence: GTTGGAAGCA, Promoter-like sequence: TATAAT, Score: -1.1192e+00
Sequence: CAAAGGAGGA, Promoter-like sequence: AATTAT, Score: -3.7729e-01
Sequence: GCAAGACATG, Promoter-like sequence: TATTTT, Score: 4.3919e-02
Sequence: AGAGATGAAA, Promoter-like sequence: TATAAT, Score: -1.1192e+00
Sequence: ATTAGAAAGT, Promoter-like sequence: TAAATT, Score: 4.2121e-01
Sequence: TTGCTGGAGC, Promoter-like sequence: TATAAT, Score: -1.1192e+00
Sequence: TTGCTGGAGC, Promoter-like sequence: TAATAT, Score: -3.7729e-01
Sequence: TCAGAAAGGA, Promoter-like sequence: TAAATT, Score: 4.2121e-01
Sequence: GACAAGGATG, Promoter-like sequence: TATAAT, Score: -1.1192e+00
Sequence: CGCAAGAAAT, Promoter-like sequence: AAAAAT, Score: 0.0000e+00
Sequence: CGCAAGAAAT, Promoter-like sequence: AAAATT, Score: 9.8083e-01
Sequence: TATTGGAGAT, Promoter-like sequence: TAAAAT, Score: -5.5962e-01
Sequence: TAGAAATCGC, Promoter-like sequence: AAAAAT, Score: 0.0000e+00
Sequence: TTTAGAAAAT, Promoter-like sequence: AAAAAT, Score: 0.0000e+00
Sequence: ATTCAGTATC, Promoter-like sequence: TATATT, Score: -1.3840e-01
Sequence: CAGAAAGGAA, Promoter-like sequence: AAAAAT, Score: 0.0000e+00
Sequence: TAAGAAGAGA, Promoter-like sequence: TAAAAT, Score: -5.5962e-01
Sequence: TATGAAGATA, Promoter-like sequence: AATTAT, Score: -3.7729e-01
Sequence: TATATGAAGA, Promoter-like sequence: AAATAT, Score: 1.8232e-01
Sequence: TATATGAAGA, Promoter-like sequence: AATATT, Score: 4.2121e-01
Sequence: TTTGAGGAGA, Promoter-like sequence: TATAAT, Score: -1.1192e+00
Sequence: GTGGAGGAAA, Promoter-like sequence: TAAAAT, Score: -5.5962e-01
Sequence: GAAATGGACC, Promoter-like sequence: TAAAAT, Score: -5.5962e-01
Sequence: GAAATGGACC, Promoter-like sequence: AAAATT, Score: 9.8083e-01
Sequence: GGAGAATGTG, Promoter-like sequence: TATTTT, Score: 4.3919e-02
Sequence: GATATTCGAT, Promoter-like sequence: AAAATT, Score: 9.8083e-01
Sequence: TTAAACCATG, Promoter-like sequence: TAAAAT, Score: -5.5962e-01
Sequence: ATTTTGTATT, Promoter-like sequence: AAATAT, Score: 1.8232e-01
Sequence: AAGGGAGAAG, Promoter-like sequence: AAAAAT, Score: 0.0000e+00
Total promoter-like sequences detected: 54
```

[54]:
```
threshold = consensus_score
promoter_presence = [any(score > threshold for score in seq.values()) for seq␣
 ↪in norm_score.values()]

present = sum(promoter_presence)
absent = len(promoter_presence) - present
```

```
plt.pie([present, absent], labels=['Promoter Detected','No Promoter'],␣
 ↪autopct='%1.1f%%', startangle=90)
plt.show()
```



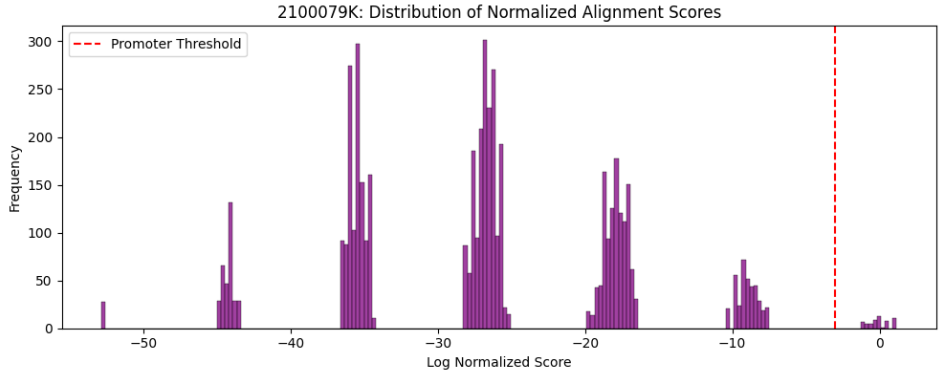
[56]:
```
def cross_validation(genome_files, gff_files, ids,ppm,consensus_score ):
    promoter_counts = []
    all_scores_list = []  # store all_scores per genome for combined histogram

    for gen_file, gff_file, id in zip(genome_files, gff_files, ids):
        print(f"Cross Validations for {id}")
        genome_record = SeqIO.read(gen_file, "fasta")
        genome_seq = str(genome_record.seq)

        # Read GFF annotation table
        gff_cols =␣
 ↪["seqid","source","type","start","end","score","strand","phase","attributes"]
        gff_df = pd.read_csv(gff_file, sep="\t", comment="#", names=gff_cols)

        print(f"for {id} Total annotated entries: {len(gff_df)}")

        # Extract promoter sequences
        promoter_seqs, _ = extracting_bases(genome_seq, gff_df)
        promoter_counts.append(len(promoter_seqs))
```

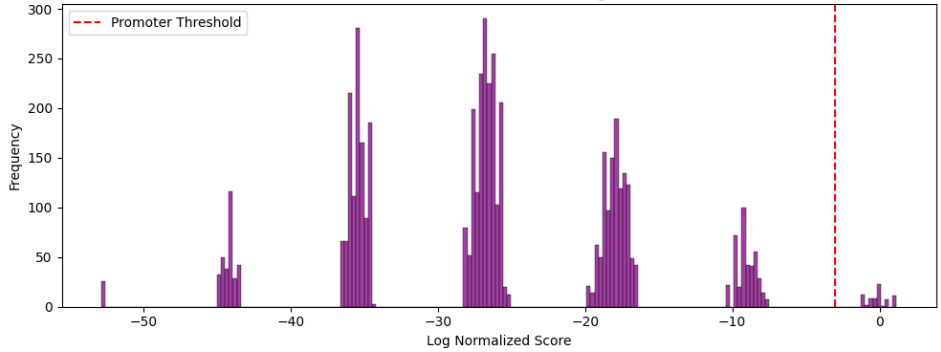
```
        # Extract upstream sequences for statistical alignment
        _, upstream_seqs_final = extracting_bases(genome_seq, gff_df)
        norm_score = statistical_alignment(upstream_seqs_final, ppm,␣
↪consensus_score)

        # Flatten scores for histogram and store
        all_scores = [score for seq_scores in norm_score.values() for score in␣
↪seq_scores.values()]
        all_scores_list.append((id, all_scores, consensus_score))

    # --- Combined histograms ---
    fig, axes = plt.subplots(len(ids), 1, figsize=(10, len(ids)*4))
    if len(ids) == 1:
        axes = [axes]  # ensure iterable

    for ax, (id, all_scores,consensus_score) in zip(axes, all_scores_list):
        sns.histplot(all_scores, bins=200, kde=False, ax=ax, color='purple')
        ax.axvline(consensus_score, color='r', linestyle='--', label='Promoter␣
↪Threshold')
        ax.set_xlabel('Log Normalized Score')
        ax.set_ylabel('Frequency')
        ax.set_title(f'{id}: Distribution of Normalized Alignment Scores')
        ax.legend()

    plt.tight_layout()
    plt.show()

    # --- Summary Visualizations ---

    # 1. Promoter counts per genome
    plt.figure(figsize=(8,4))
    plt.bar(ids, promoter_counts, color='skyblue')
    plt.ylabel("Predicted promoter sequences")
    plt.title("Cross-validation: Number of predicted promoters per genome")
    plt.show()
```

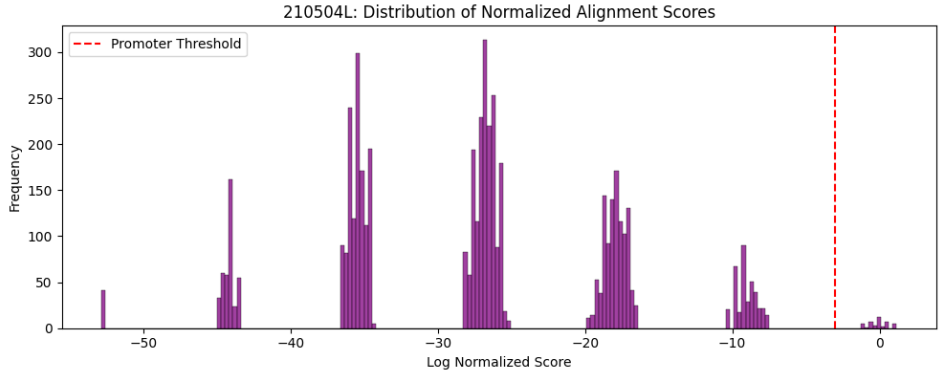
[57]:
```
#cross validation
genome_files = ['NCBI_Data/210079K_GCA_001457635.1/ncbi_dataset/data/
↪GCA_001457635.1/GCA_001457635.1_NCTC7465_genomic.fna',
                'NCBI_Data/210179R_GCA_019048645.1/ncbi_dataset/data/
↪GCA_019048645.1/GCA_019048645.1_ASM1904864v1_genomic.fna',
                'NCBI_Data/210504L_GCA_900636475.1/ncbi_dataset/data/
↪GCA_900636475.1/GCA_900636475.1_42197_F01_genomic.fna',
                'NCBI_Data/210657G_GCA_900637025.1/ncbi_dataset/data/
↪GCA_900637025.1/GCA_900637025.1_46338_H01_genomic.fna',
```

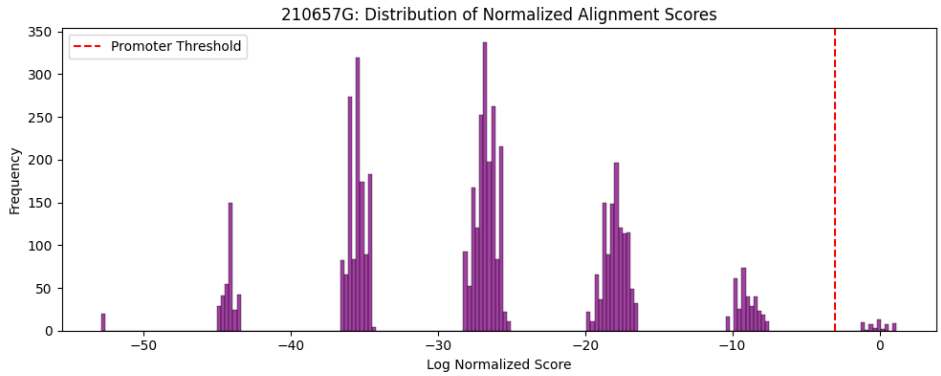
```
                'NCBI_Data/210707L_GCA_900475505.1/ncbi_dataset/data/
 ↪GCA_900475505.1/GCA_900475505.1_42925_G01_genomic.fna']

gff_files = ['NCBI_Data/210079K_GCA_001457635.1/ncbi_dataset/data/GCA_001457635.
 ↪1/genomic.gff',
             'NCBI_Data/210179R_GCA_019048645.1/ncbi_dataset/data/GCA_019048645.
 ↪1/genomic.gff',
             'NCBI_Data/210504L_GCA_900636475.1/ncbi_dataset/data/GCA_900636475.
 ↪1/genomic.gff',
             'NCBI_Data/210657G_GCA_900637025.1/ncbi_dataset/data/GCA_900637025.
 ↪1/genomic.gff',
             'NCBI_Data/210707L_GCA_900475505.1/ncbi_dataset/data/GCA_900475505.
 ↪1/genomic.gff']

ids = ['2100079K','210179R','210504L','210657G','210707L']

cross_validation(genome_files,gff_files,ids, ppm,consensus_score)
```

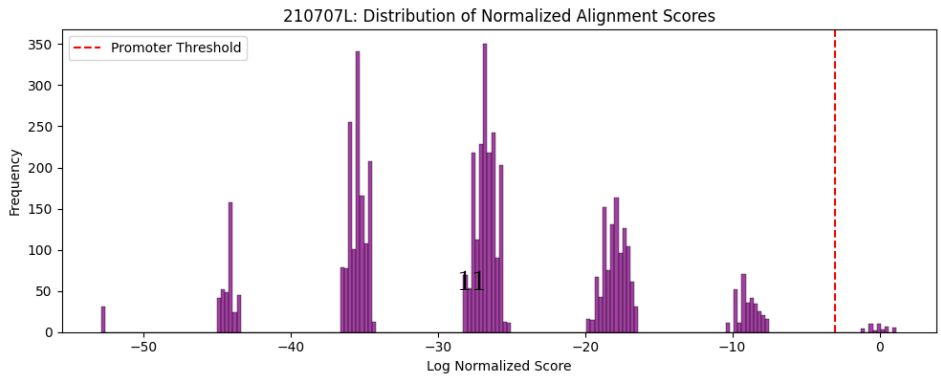
```
Cross Validations for 2100079K
for 2100079K Total annotated entries: 4354
Cross Validations for 210179R
for 210179R Total annotated entries: 4080
Cross Validations for 210504L
for 210504L Total annotated entries: 3939
Cross Validations for 210657G
for 210657G Total annotated entries: 3795
Cross Validations for 210707L
for 210707L Total annotated entries: 4565
```

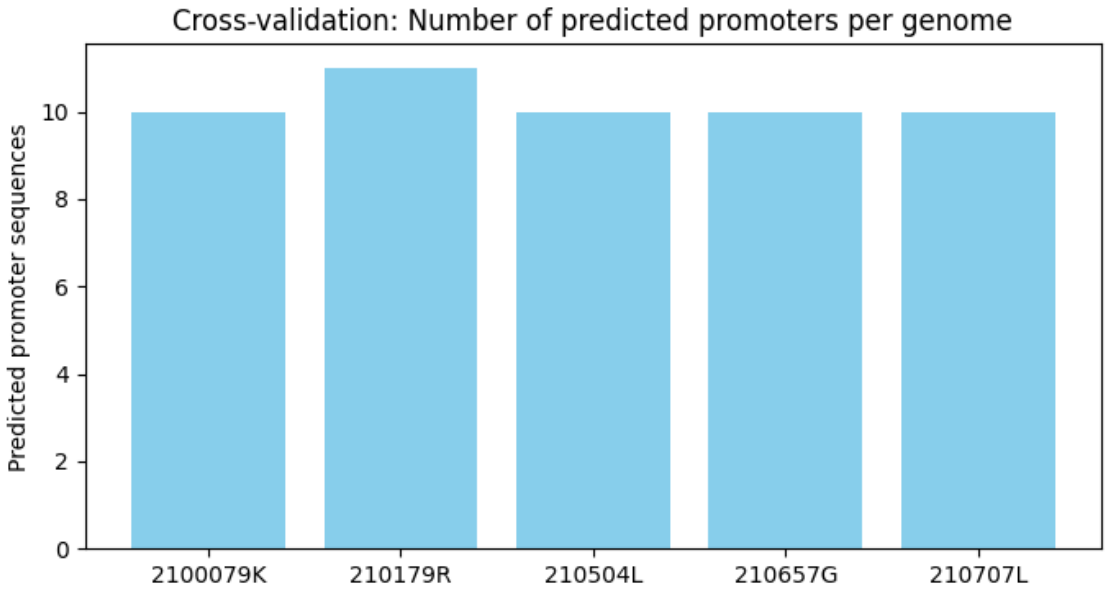

2100079K: Distribution of Normalized Alignment Scores
Promoter Threshold
Frequency
Log Normalized Score


210179R: Distribution of Normalized Alignment Scores
Promoter Threshold
Frequency
Log Normalized Score


210504L: Distribution of Normalized Alignment Scores
Promoter Threshold
Frequency
Log Normalized Score


210657G: Distribution of Normalized Alignment Scores
Promoter Threshold
Frequency
Log Normalized Score


210707L: Distribution of Normalized Alignment Scores
Promoter Threshold
Frequency
Log Normalized Score


Cross-validation: Number of predicted promoters per genome
Predicted promoter sequences
0
2
4
6
8
10
2100079K
210179R
210504L
210657G
210707L

[ ]: