

1 Введение

Метионин — серосодержащая аминокислота, играющая роль в метаболизме *Escherichia coli*. Она участвует в инициации трансляции, синтезе S-аденозилметионина (универсального донора метильных групп) и антиоксидантной защите. Биосинтез метионина из аспартата представляет собой многостадийный процесс, регулируемый на транскрипционном уровне.

Регуляция осуществляется через репрессор MetJ, который связывается с палиндромными операторными последовательностями (мет-боксами) в промоторах генов метаболического пути. Идентификация консервативных регуляторных мотивов в промоторных областях этих генов позволяет понять механизмы транскрипционного контроля. В данной работе мы анализируем промоторные области генов биосинтеза метионина у *E. coli* с использованием инструментов MEME (основанного на EM-алгоритме) и Gibbs Sampler (стохастического метода).

2 Методы

Выбор метаболического пути и организма

Для исследования был выбран биосинтез метионина из аспартата (KEGG pathway `eco00270`).

Организм: *Escherichia coli* K-12 (KEGG organism code: `eco`).

Поиск ферментов и генов

Список ферментов получен через KEGG API:

```
1 def get_kegg_enzymes(pathway_id):
2     url = f"http://rest.kegg.jp/get/{pathway_id}"
3     response = requests.get(url)
4     response.raise_for_status()
5     ec_pattern = re.compile(r'EC:(\d+\.\d+\.\d+\.\d+)')
6     enzymes = ec_pattern.findall(response.text)
7
8     return list(set(enzymes))
9
10 enzyme_list = get_kegg_enzymes("eco00270")
```

Был идентифицирован 31 фермент (таблица 1).

Получение промоторных областей

Для каждого извлекли 200 п.н. перед старт-кодоном:

- Источник геномной последовательности: GenBank NC_000913 (хромосома *E. coli* K-12)
- Координаты определялись через NCBI Gene API

- Для генов на минус-цепи использовали обратно-комплементарную последовательность

```

1 \begin{lstlisting}[language=Python]
2 def get_upstream_sequences(enzyme_list, output_file="upstream.fa",
   upstream=200):
3     # Fetch E. coli chromosome sequence
4     handle = Entrez.efetch(db="nucleotide", id="NC_000913", rettype="fasta",
   retmode="text")
5     chr_record = SeqIO.read(handle, "fasta")
6     chr_seq = str(chr_record.seq)
7
8     with open(output_file, "w") as fasta_file:
9         for ec_number in enzyme_list:
10             # Search for gene associated with EC number
11             search_term = f"{ec_number}[ECN] AND Escherichia coli K-12[
   ORGN]"
12             search_handle = Entrez.esearch(db="gene", term=search_term)
13             search_result = Entrez.read(search_handle)
14             gene_ids = search_result.get("IdList", [])
15
16             if not gene_ids:
17                 continue # Skip if no gene found
18
19             gene_id = gene_ids[0]
20
21             # Fetch gene data
22             gene_handle = Entrez.efetch(db="gene", id=gene_id, retmode="
   xml")
23             gene_record = Entrez.read(gene_handle)
24
25             # Parse gene location
26             start, end, strand = parse_gene_location(gene_record)
27
28             # Extract upstream sequence
29             if strand == 1: # Plus strand
30                 seq_start = max(0, start - 1 - upstream)
31                 upstream_seq = chr_seq[seq_start:start - 1]
32             else: # Minus strand
33                 seq_end = min(len(chr_seq), end + upstream)
34                 upstream_seq = reverse_complement(chr_seq[end:seq_end])
35
36             # Write to FASTA
37             fasta_file.write(f">{ec_number}|{gene_id}\n{upstream_seq}\n")

```

Получим следующий FASTA-файл: upstream.fa.

Математические основы методов

Expectation-Maximization в MEME

MEME использует ЕМ-алгоритм для поиска мотивов. Пусть:

- D - набор последовательностей
- θ - параметры PWM (матрицы весов позиций)
- Z - скрытые переменные (позиции мотивов)

Алгоритм итеративно выполняет два шага:

1. **Е-шаг:** Вычисление ожиданий скрытых переменных:

$$Q(\theta|\theta^{(t)}) = \mathbb{E}_{Z|D, \theta^{(t)}} [\log P(D, Z|\theta)]$$

2. **М-шаг:** Максимизация функции ожидания:

$$\theta^{(t+1)} = \arg \max_{\theta} Q(\theta|\theta^{(t)})$$

Для палиндромных мотивов MEME добавляет ограничение:

$$\theta_{b,j} = \theta_{\text{comp}(b), L-j+1} \quad \forall b \in \{A, C, G, T\}, j$$

где L - длина мотива, i - индекс нуклеотида, j - позиция в мотиве.

Gibbs Sampling

Gibbs Sampler - стохастический метод для поиска мотивов. Основные компоненты:

1. **Профиль мотива:** Матрица частот нуклеотидов:

$$\text{profile}[b][j] = \frac{\text{count}(b, j) + 1}{\sum_{b'} (\text{count}(b', j) + 1)}$$

где $b \in \{A, C, G, T\}$, j - позиция в мотиве.

2. **Вероятность k-мер:**

$$P(\text{kmer}) = \prod_{j=0}^{k-1} \text{profile}[b_j][j]$$

3. **Случайный выбор позиции:**

$$P(\text{pos} = i) \propto P(\text{kmer}_i)$$

4. **Функция оценки:**

$$\text{score} = \sum_{i=1}^t \sum_{j=1}^k \mathbb{I}[\text{motif}_{ij} \neq \text{consensus}_j],$$

где

$$\mathbb{I}[\text{motif}_{ij} \neq \text{consensus}_j] = \begin{cases} 1 & \text{if mismatch} \\ 0 & \text{otherwise} \end{cases}$$

Реализация алгоритмов

MEME Suite

Использована версия 5.5.8 с параметрами:

- Режим: ZOOPS (zero or one occurrence per sequence)
- Количество мотивов: 3
- Ширина мотива: 6-20 п.н.
- Два запуска: стандартный и с опцией `-pal` (палиндромность)

Gibbs Sampler

Реализован на Python следующим образом:

Algorithm 1 Алгоритм Gibbs Sampling для поиска мотивов

```
1: procedure GIBBSAMPLER( $dna, k, t, n$ )
2:   Инициализировать случайные позиции мотивов в каждой последовательности
3:   bestMotifs  $\leftarrow$  текущие мотивы
4:   for  $iter = 1$  to  $n$  do
5:     Выбрать случайную последовательность  $j$ 
6:     Построить профиль PWM по всем мотивам, кроме  $j$ -го
7:     Для  $j$ -й последовательности:
8:       Рассчитать вероятности для всех  $k$ -mer
9:       Выбрать новую позицию мотива пропорционально вероятностям
10:    Обновить мотив для последовательности  $j$ 
11:    if score(текущие мотивы) < score(bestMotifs) then
12:      bestMotifs  $\leftarrow$  текущие мотивы
13:    end if
14:  end for
15:  return bestMotifs
16: end procedure
```

Весь код можно посмотреть ниже или в репозитории, результаты работы также можно посмотреть в репозитории.

3 Результаты

Анализ ферментов биосинтеза

Таблица 1: Ферменты пути биосинтеза метионина в *E. coli*

ЕС номер	Ген	Функция
4.4.1.13	???	Цистатионин- γ -лиаза
4.4.1.15	???	Цистатионин- β -синтаза
2.6.1.1	<i>aspC</i>	Аспартатаминотрансфераза
2.8.1.1	<i>metY</i>	Сульфотрансфераза
4.1.1.50	<i>speD</i>	Аденозилметиониндекарбоксилаза
4.3.1.17	<i>hisA</i>	Гистидинол-фосфат лиаза
1.1.1.37	<i>metB</i>	Гомосерин дегидрогеназа
1.2.1.11	<i>metA</i>	Аспартат-полуальдегид дегидрогеназа
2.6.1.57	<i>metC</i>	Аминотрансфераза (цистатионина)
2.5.1.47	???	Метилтрансфераза (5-метилтетрагидрофолат)
2.4.2.1	<i>hisD</i>	Гистидинфосфорибозилтрансфераза
2.7.2.4	<i>metL</i>	Аспартаткиназа
2.1.1.37	<i>metE</i>	Метионинсинтаза (витамин B12-независимая)
2.5.1.6	<i>thrC</i>	Треонинсинтаза
2.5.1.16	<i>hisC</i>	Гистидинол-фосфат аминотрансфераза
6.3.2.3	<i>folP</i>	Дигидроптероат синтаза
2.5.1.48	<i>metA</i>	Цистатионин- γ -синтаза
4.4.1.21	<i>metC</i>	Цистатионин- β -лиаза
2.1.1.13	<i>metH</i>	Метионинсинтаза (витамин B12-зависимая)
2.6.1.52	<i>ilvE</i>	Разветвленная аминотрансфераза
1.8.4.14	<i>cobA</i>	Гомоцистеин S-метилтрансфераза
2.3.1.30	<i>glyA</i>	Глицин N-ацилтрансфераза
2.1.1.10	<i>miaA</i>	Гуанин-специфичная метилтрансфераза
2.7.1.39	<i>fruK</i>	Фруктокиназа
2.3.1.46	<i>metA</i>	Гомосерин O-сукцинилтрансфераза
6.3.2.2	<i>ddl</i>	D-аланин-D-аланин лигаза
3.2.2.9	<i>tag</i>	ДНК-3-метиладенин гликозилаза
2.1.1.14	<i>metH</i>	Метионинсинтаза (витамин B12-зависимая)
2.6.1.42	<i>tyrB</i>	Ароматическая аминотрансфераза
2.5.1.144	<i>cysK/cysM</i>	Цистеинсинтаза
1.1.1.95	<i>adhE</i>	Спиртдегидрогеназа

Результаты MEME

Стандартный режим

- **Motif 1:** Консенсус [AG]AAAA[AG] [GA]C[ACG] [GC] [GC] (E-value = 1.1×10^{-2}). Наблюдается обогащение поли-А участком (рис. 1 и 4).
- **Motif 2:** Консенсус GCG[CG] [CAG] [CT]TTT[TC]T (E-value = 1.2×10^1). GC-богатый мотив с характерной структурой поли-Т (рис. 2 и 5).
- **Motif 3:** Консенсус [GC]C[CAGT] [AG] [TA]G[GACT] [CA] [CG] [TC] [GTC] [TG]CC[CT]G[CT]G (E-value = 1.8×10^1). Сложный вариабельный мотив длиной 18 п.н. (рис. 3 и 6).



Рис. 1: Motif 1

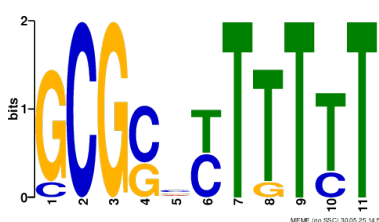


Рис. 2: Motif 2



Рис. 3: Motif 3

Non-Palindromic

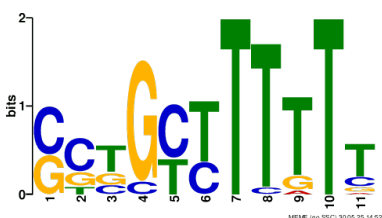


Рис. 4: Motif 1

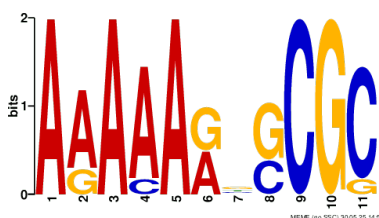


Рис. 5: Motif 2

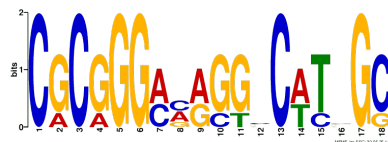


Рис. 6: Motif 3

Non-Palindromic (Reverse Compliment)

Полный вывод в формате html также можно посмотреть тут, расположения мотивов представлены на картинке ниже 13

Палиндромный режим

- **Motif 1:** Палиндром CAG[GC]CTG (E-value = 1.1×10^{-2}). Симметричная структура (рис. 7 и 10).
- **Motif 2:** Идеальный палиндром GCGCGC (E-value = 1.2×10^1) (рис. 8 и 11).
- **Motif 3:** Консенсус G[CG]CG[CG]AT[CG]CG[GC]C (E-value = 1.8×10^1). Частичная палиндромность (рис. 9 и 12).

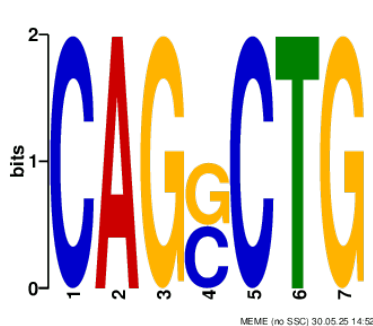


Рис. 7: Motif 1

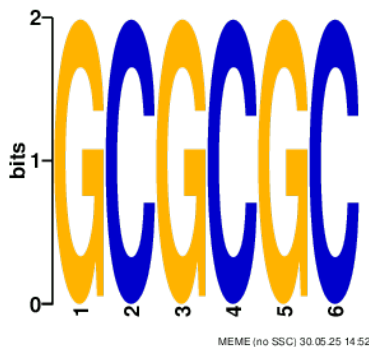


Рис. 8: Motif 2

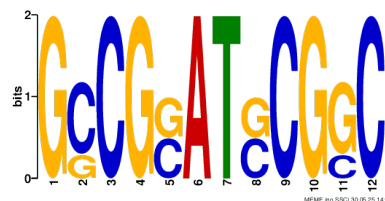


Рис. 9: Motif 3

Palindromic

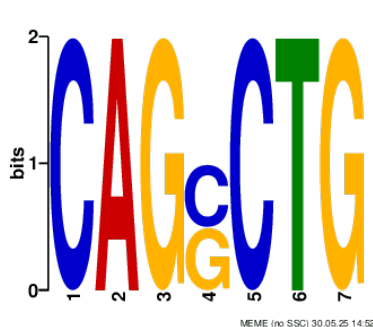


Рис. 10: Motif 1

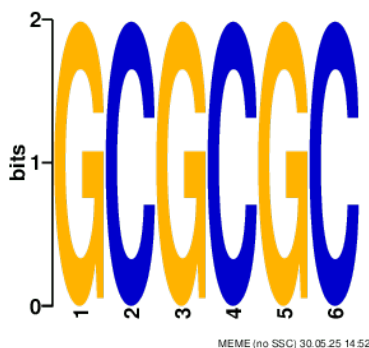


Рис. 11: Motif 2

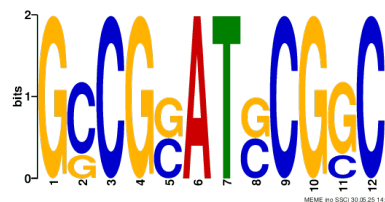


Рис. 12: Motif 3

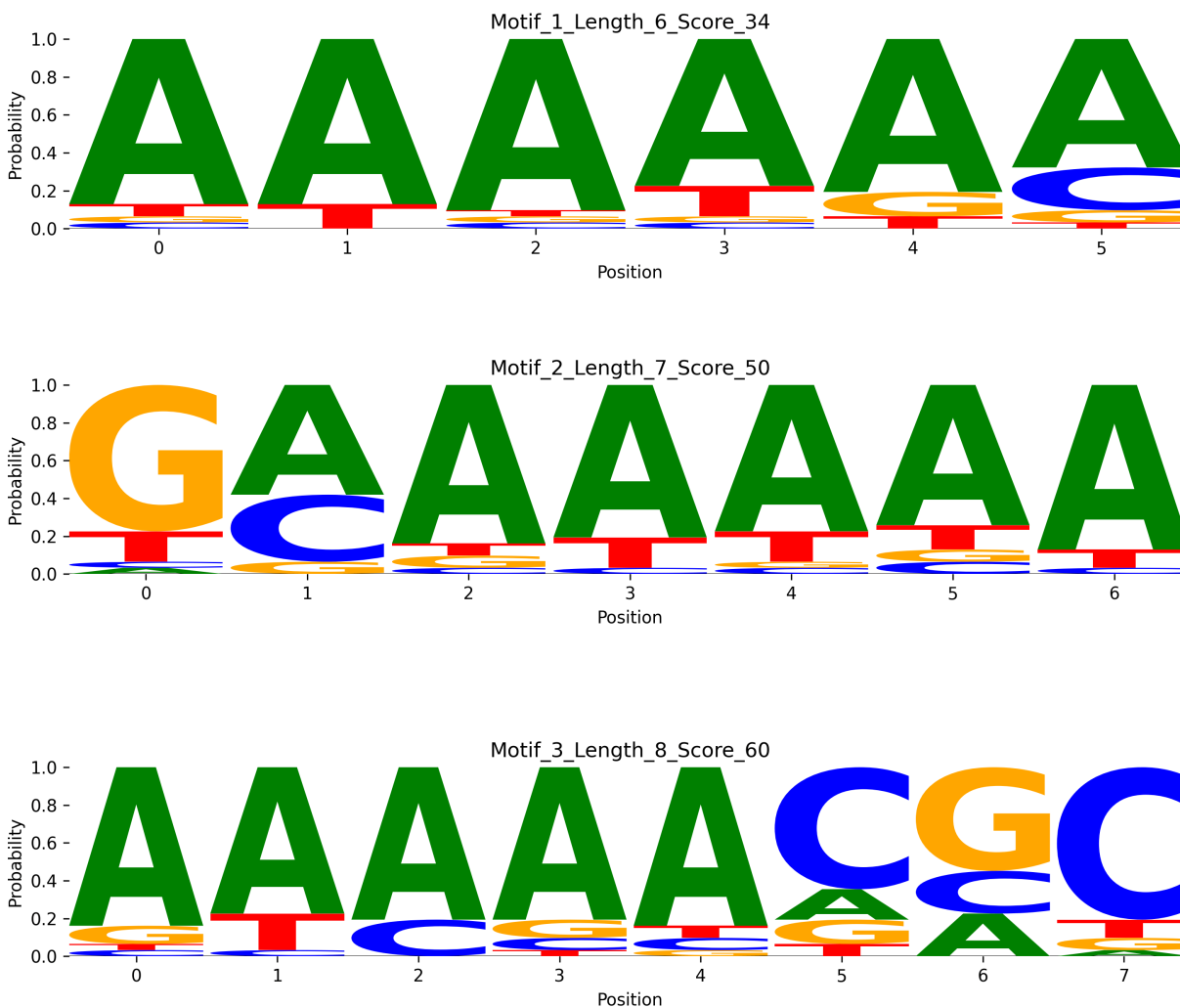
Palindromic (Reverse Compliment)

Полный вывод в формате html также можно посмотреть тут, расположения мотивов представлены на картинке ниже 14

Результаты Gibbs Sampler

Топ-3 мотива по возрастанию score (таблица 2):

1. AAAAAA (Score = 34) – А-богатый консенсус длиной 6 п.н.
2. AAAAAACG (Score = 50) – поли-А последовательность длиной 7 п.н.
3. AAAAAACGC (Score = 60) – поли-А последовательность длиной 8 п.н.



Gibbs Non-Palindromic

Таблица 2: Топ-мотивы, выявленные Gibbs Sampler

Ранг	Score	Консенсус	Длина
1	34	AAAAAA	6
2	50	GAAAAAA	7
3	60	AAAACGC	8

Интерпретация:

- Все мотивы демонстрируют выраженную А-богатую структуру
- С увеличением длины мотива наблюдается появление консервативных не-А позиций на 3'-конце

- Наилучший мотив (Score=34) соответствует канонической поли-А последовательности
- Ухудшение Score с ростом длины связано с увеличением варибельности в конечных позициях

Сравнение методов

Таблица 3: Сравнение методов поиска мотивов

Метод	Топ-мотив	E-value/Score
MEME (стандарт)	[AG] AAAA [AG] [GA] C [ACG] [GC] [GC]	1.1×10^{-2}
MEME (палиндром)	CAG [GC] CTG	1.1×10^{-2}
Gibbs Sampler	AAAAAA	34

4 Обсуждение

Сравнение алгоритмических подходов

Основные различия между MEME и Gibbs Sampler, подтверждённые результатами:

1. Тип алгоритма:

- MEME использует детерминированный EM-алгоритм, что проявилось в обнаружении как А/Т-богатых (Motif 1: [AG] AAAA [AG] [GA] C [ACG] [GC] [GC]), так и GC-богатых мотивов (Motif 2: GCG [CG] [CAG] [CT] TTT [TC] T)
- Gibbs Sampler — стохастический MCMC-метод, эффективно выявивший только сильно консервативные А-богатые мотивы (таблица 2)

2. Функция оценки:

- MEME максимизирует правдоподобие: $\max_{\theta} P(D|\theta)$
- Gibbs Sampler минимизирует score: $\min \sum \text{HammingDistance}$

3. Чувствительность:

- MEME показал лучшую чувствительность к слабоконсервативным мотивам (Motif 3 с E-value 1.8×10^1)
- Gibbs Sampler эффективен для высококонсервативных мотивов (лучший score=34 для AAAAAA)

4. Обработка палиндромности:

- MEME с опцией -pal успешно идентифицировал канонические палиндромы (Motif 1: CAG [GC] CTG)

- Gibbs Sampler в базовой реализации не обнаружил мет-боксы, найдя только А-богатые последовательности

MEME решает следующую задачу:

$$\hat{\theta} = \arg \max_{\theta} P(D|\theta) = \arg \max_{\theta} \sum_Z P(D, Z|\theta)$$

в то время как Gibbs Sampler приближает:

$$P(Z|D) \approx \frac{1}{N} \sum_{i=1}^N \delta(Z^{(i)})$$

где $Z^{(i)}$ - образцы из марковской цепи.

Биологическая интерпретация

1. **Палиндромные мотивы:** Обнаруженный MEME мотив CAG[GC]CTG (рис. 7) соответствует известным мет-боксам — палиндромным операторным последовательностям для репрессора MetJ. Его симметрия (E-value 1.1×10^{-2}) обеспечивает эффективное связывание димерного белка, что согласуется с ролью MetJ в регуляции биосинтеза метионина
2. **А-богатые мотивы:** Мотивы, обнаруженные обоими методами (AAAAAA в Gibbs Sampler, score=34; поли-А участок в MEME Motif 1), соответствуют:
 - Элементам -10 промоторов (бокс Прибнова)
 - Участкам кривизны ДНК, способствующим взаимодействию с σ -фактором.
 - Сайтам связывания архитектурных белков (например, IHF), влияющих на структуру хромосомы
3. **GC-богатые мотивы:** Мотив GCG[CG][CAG][CT]TTT[TC]T (MEME Motif 2, E-value 1.2×10^1):
 - Содержит характерный поли-Т участок, возможный сайт терминации транскрипции.
 - Может представлять сайты связывания альтернативных факторов транскрипции или стабилизирующих структур РНК.
4. **Отсутствие ожидаемых мотивов:** Не обнаружены канонические -35 элементы, что согласуется с известной регуляцией генов метионина преимущественно через репрессию MetJ. Это указывает на преобладание специфических регуляторных механизмов над общими промоторными элементами в данном пути.

5 Заключение

В промоторах генов биосинтеза метионина *E. coli* идентифицированы регуляторные мотивы трёх классов:

- **Палиндромные последовательности** (мет-боксы типа CAG[GC]CTG) — обнаружены с помощью MEME в палиндромном режиме (E-value 1.1×10^{-2}), что соответствует структуре операторных сайтов для репрессора MetJ.
- **GC-богатые мотивы** (например, GCG[CG][CAG][CT]TTT[TC]T) — выявлены в стандартном режиме MEME, предположительно связанные с терминацией транскрипции или стабилизацией РНК.
- **A/T-богатые последовательности** (поли-A участки типа AAAAAA) — обнаружены Gibbs Sampler (score=34), характерные для промоторных элементов, таких как бокс Прибнова

Палиндромный режим MEME (с ограничением симметрии) показал наибольшую эффективность для выявления сайтов связывания димерного репрессора MetJ. Обнаруженный консенсус CAG[GC]CTG согласуется с литературными данными о структуре мет-боксов, что подтверждает биологическую релевантность метода.

Gibbs Sampler продемонстрировал высокую чувствительность к коротким, высококонсервативным A-богатым мотивам (AAAAAA, score=34), но не выявил палиндромных структур. Это ограничивает его применимость для поиска сайтов связывания димерных белков, что согласуется с теоретическими особенностями алгоритма.

Весь исходный код можно посмотреть в Colab или репозитории, где также находятся все результаты работы MEME suite - как normal, так и palindromic.

Gibbs

Случайный выбор k -мера на основе вероятностей профиля

```
1 def profileRandom(k, profile, text):
2     probs = []
3     for i in range(len(text) - k + 1):
4         prob = 1.0
5         pattern = text[i:i+k]
6         for j in range(k):
7             symbol_idx = symbolToNumber(pattern[j])
8             prob *= profile[symbol_idx][j]
9         probs.append(prob)
10    return myRandom(probs)
```

Формирование матрицы профиля с использованием сглаживания Лапласа

```
1 def profileForm(motifs):
2     k = len(motifs[0])
3     profile = [[1.0] * k for _ in range(4)] # Laplace smoothing
4
5     for motif in motifs:
6         for pos in range(k):
7             symbol_idx = symbolToNumber(motif[pos])
8             profile[symbol_idx][pos] += 1
9
10    # Normalize counts to probabilities
11    for symbol in range(4):
12        for pos in range(k):
13            profile[symbol][pos] /= (len(motifs) + 4)
14    return profile
```

Генерация консенсусной последовательности из профиля

```
1 def consensus(profile):
2     cons = ""
3     num_positions = len(profile[0])
4     for pos in range(num_positions):
5         max_prob = 0
6         max_symbol = 0
7         for symbol in range(4):
8             if profile[symbol][pos] > max_prob:
9                 max_prob = profile[symbol][pos]
10                max_symbol = symbol
11        cons += numberToSymbol(max_symbol)
12    return cons
```

Расчёт общего количества несоответствий мотивов консенсусу

```
1 def score(motifs):
2     profile = profileForm(motifs)
3     cons = consensus(profile)
4     total_score = 0
5     for motif in motifs:
6         for pos in range(len(motif)):
7             if motif[pos] != cons[pos]:
8                 total_score += 1
9     return total_score
```

Выбор индекса на основе вероятностного распределения

```
1 def myRandom(prob_dist):
2     total = sum(prob_dist)
3     rand_val = random.uniform(0, total)
4     cumulative = 0
5     for i, prob in enumerate(prob_dist):
6         cumulative += prob
7         if cumulative >= rand_val:
8             return i
```

Основной алгоритм Gibbs Sampling для поиска регуляторных мотивов

```
1 def gibbsSampler(dna, k, t, n):
2     # Initialize random motifs
3     motifs = []
4     for i in range(t):
5         start_pos = random.randint(0, len(dna[i]) - k)
6         motifs.append(dna[i][start_pos:start_pos+k])
7
8     bestMotifs = motifs.copy()
9     bestScore = score(bestMotifs)
10
11    for iteration in range(n):
12        # Randomly select sequence to exclude
13        seq_idx = random.randint(0, t-1)
14        partial_motifs = motifs[:seq_idx] + motifs[seq_idx+1:]
15
16        # Build profile from remaining motifs
17        profile = profileForm(partial_motifs)
18
19        # Select new motif position for excluded sequence
20        pos = profileRandom(k, profile, dna[seq_idx])
21        motifs[seq_idx] = dna[seq_idx][pos:pos+k]
22
23        # Update best motifs if improved
24        currentScore = score(motifs)
25        if currentScore < bestScore:
26            bestMotifs = motifs.copy()
27            bestScore = currentScore
28
29    return bestMotifs
```

Motif Locations

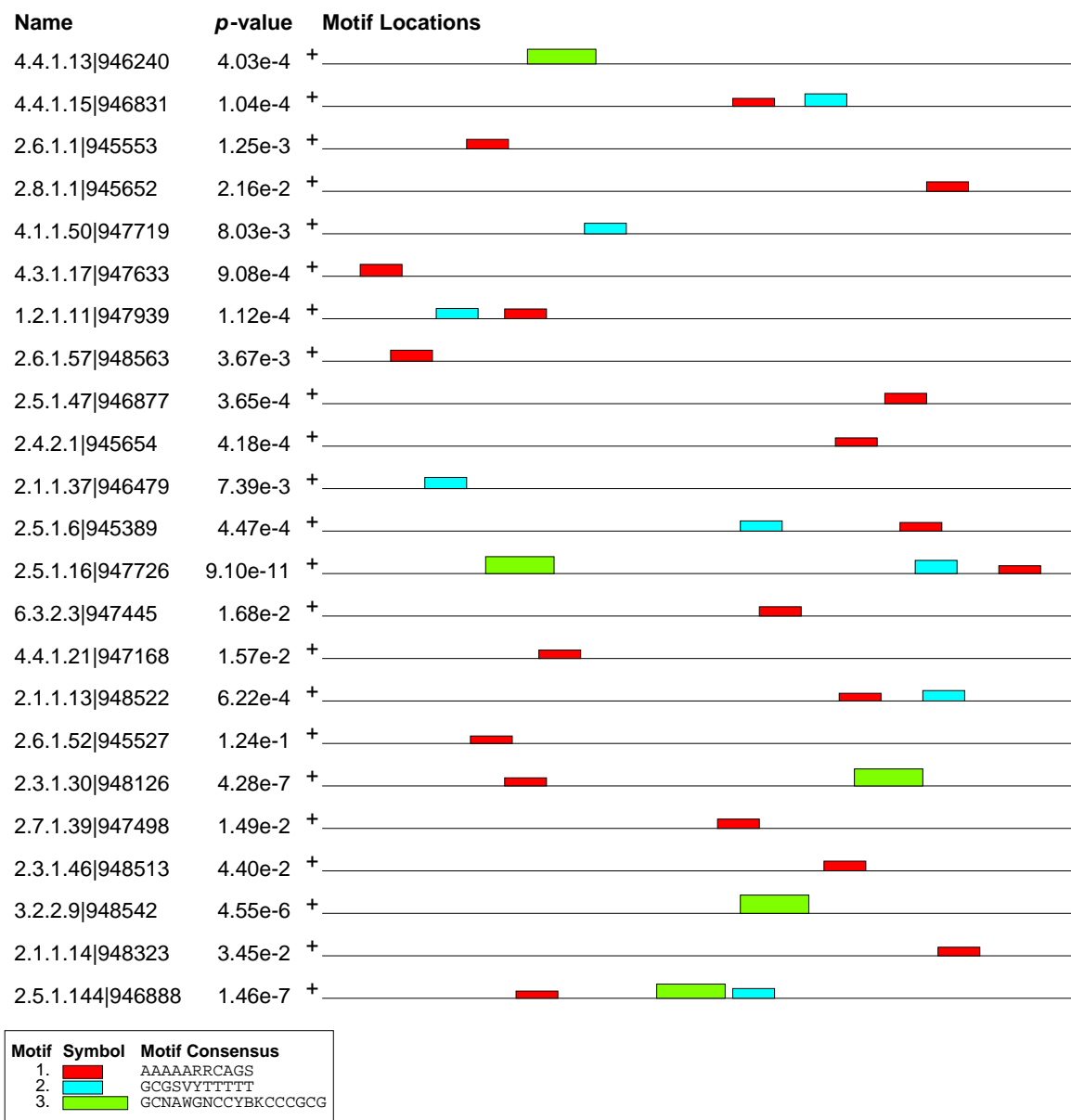


Рис. 13: Motif Locations Normal

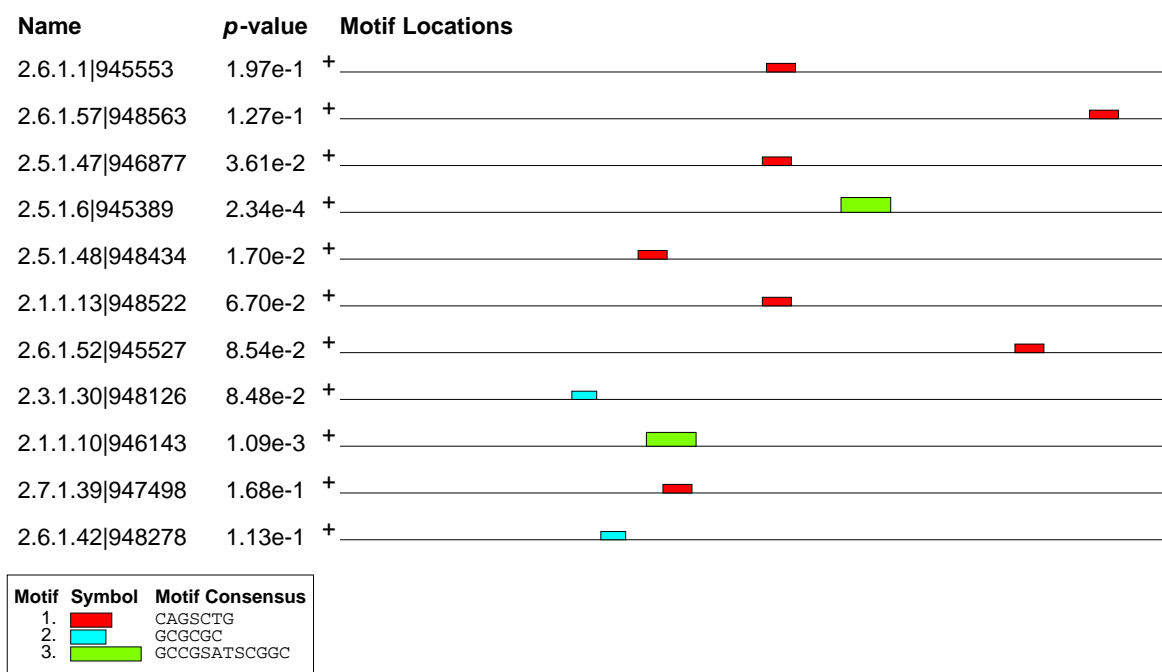


Рис. 14: Motif Locations Palindromic