

Нанопоровое секвенирование в геномике, метагеномике и эпигеномике: инструменты и алгоритмы анализа

Ерёмин А.А.*, Сергеев А.В., Зверева М.Э., Хренова М.Г.

Московский государственный университет имени М.В.Ломоносова, Москва, Россия

Аннотация. Нанопоровое секвенирование представляет собой перспективную технологию определения последовательности фрагментов ДНК и РНК. В данной работе рассматриваются современные алгоритмы и программные инструменты, применяемые для анализа данных нанопорового секвенирования в геномике, метагеномике и эпигеномике. В статье описаны алгоритмы наиболее распространенных программных инструментов, включая их преимущества и ограничения. Представленный обзор полезен для исследователей, занимающихся анализом данных нанопорового секвенирования, а также для разработчиков новых методов обработки биоинформатических данных.

Ключевые слова: нанопоровое секвенирование, сборка генома *de novo*, геномика, эпигеномика, метагеномика, длинные прочтения

ВВЕДЕНИЕ

Развитие технологий секвенирования за последние десятилетия произвело революцию в области биологии. Методы секвенирования второго поколения (секвенирование путем синтеза [1] на платформах Illumina, Ion Torrent, BGI, Roche, GeneMind, Salus) позволяют получать обширные наборы данных с высокой точностью и сравнительно низкой себестоимостью, однако имеют ограничения в длине прочтений (обычно 300 пар оснований) и не позволяют изучать химические модификации нуклеиновых кислот в полном объеме [2, 3]. Стало очевидно, что для решения ряда задач – в частности, связанных с анализом структурных перестроек генома, идентификацией новых вариантов сплайсинга или сложных метагеномных сообществ – требуются методы, позволяющие получать более длинные прочтения в десятки и сотни тысяч пар оснований [4, 5, 6]. Наиболее перспективными в этом отношении оказались технологии третьего поколения, а именно - платформы нанопорового секвенирования [7] благодаря самой низкой себестоимости.

Нанопоровое секвенирование характеризуется рядом уникальных свойств, выгодно отличающих его от методов предыдущих поколений [8]. Во-первых, оно обеспечивает чтение сверхдлинных фрагментов ДНК, тысячи и сотни тысяч пар оснований. Рекорд по длине – более четырех миллионов пар оснований [9]. Это позволяет более точно восстанавливать структуру генома и выявлять крупные структурные вариации. Во-вторых, нанопоровые технологии дают возможность прямого считывания модификаций нуклеотидных остатков, что открывает новые возможности в изучении механизмов эпигенетической регуляции. В-третьих, портативность [10, 11] и относительно невысокая стоимость нанопоровых секвенаторов существенно расширяют

*ereminaa@my.msu.ru

сферу их применения – от полевых исследований и клинических анализов до научных лабораторий.

Однако, несмотря на явные преимущества, обработка и анализ данных нанопорового секвенирования в различных омиксных экспериментах (геномных, метагеномных, эпигеномных и др.) остаются не стандартизированной задачей [12]. Ошибки при секвенировании, точность предварительной обработки сигналов, неоднородное покрытие, необходимость корректного распознавания модифицированных нуклеотидных остатков и учёта высокого уровня шума данных – всё это требует разработки специализированных конвейеров и алгоритмов [13]. На сегодняшний день существует множество программных инструментов для различных стадий анализа (выравнивание, сборка контигов, идентификация вариаций, детекция метилирования и т. д.), однако единых, универсальных и общепринятых решений, как в случае с секвенированием второго поколения [14, 15, 16], пока не сформировано. Более того, высокие темпы появления новых версий программ и растущая сложность используемых алгоритмов затрудняют выбор оптимальных методов и, как следствие, воспроизводимость экспериментов [17].

В рамках метагеномных исследований актуальность использования нанопорового секвенирования объясняется возможностью быстрого и полного описания микробных сообществ, включая редко встречающиеся виды или штаммы, которые остаются «невидимыми» для методов секвенирования второго поколения [18]. Вместе с тем, эпигеномные исследования методом нанопорового секвенирования выигрывают за счёт прямой регистрации эпигенетических модификаций, которые крайне важны для понимания регуляции экспрессии генов и механизмов заболеваний [19]. В геномных исследованиях длинные прочтения чрезвычайно полезны при реконструкции сложных геномных областей, содержащих tandemные повторы, а также выявлении транслокаций, инверсий и других структурных вариаций [20]. Однако для всего этого необходимы специфические конвейеры (последовательность программных инструментов, предназначенных для обработки, анализа и интерпретации данных), позволяющие учитывать особенности качества, длины и специфики данных, а также тонкости в использовании конкретных биоинформатических инструментов.

Таким образом, целью этого обзора является систематизация современных алгоритмов и программных решений, применяемых при анализе данных нанопорового секвенирования в геномике, метагеномике и эпигеномике. Мы последовательно рассмотрим разные этапы обработки данных с акцентом на ключевые математические подходы, метрики качества и практические ограничения, и предложим свою схему обработки данных нанопорового секвенирования ([?]).

1. КОНВЕЙЕР ОБРАБОТКИ НАНОПОРОВЫХ ДАННЫХ

1.1. Декодирование

Нанопоровый секвенатор записывает прочтения в виде изменения ионного тока, проходящего через нанопору в момент, когда молекула нуклеиновой кислоты проходит через неё. Нанопора – это молекулярное отверстие наноразмера (диаметром порядка нескольких нанометров), встроенное в тонкую мембрану, через которую пропускается одиночная молекула ДНК или РНК. Мембрана отделяет два резервуара с электролитом, и при подаче напряжения через нанопору создается ионный ток. Когда молекула нуклеиновой кислоты начинает проходить через нанопору под действием электрического поля, она частично блокирует ток, вызывая характерные изменения его амплитуды. Эти изменения зависят от последовательности нуклеотидных остатков, находящихся в данный момент внутри поры. Таким образом, каждый фрагмент цепи вызывает

уникальный сигнал, который затем интерпретируется для восстановления исходной последовательности нуклеиновой кислоты [21]. Полученные данные фиксируются в виде так называемых «сырых сигналов» – временных рядов значений ионного тока. Эти сигналы записываются в бинарных форматах, таких как fast5 (на основе HDF5 [22]) или его более производительном и параллельно-ориентированном преемнике pod5, который предназначен для более эффективной обработки и хранения большого объема данных. Эти форматы содержат не только сигналы, но и сопутствующую информацию: метаданные эксперимента, данные о канале, времени, настройках прибора.

Pod5 файлы [23] – это новый формат файлов, который постепенно заменяет fast5 формат. Pod5 файлы решают некоторые ограничения и проблемы, имеющиеся у fast5 формата, и обладают большей производительностью как в плане скорости чтения/записи данных, так и в плане их обработки. Это особенно важно при работе с большими объемами данных, характерными для современных омиксных экспериментов. Новый формат pod5 также предлагает более оптимизированную структуру данных для упрощенного доступа к различным типам информации и повышает общую эффективность работы с файлами. Pod5 файлы совместимы со всем программным обеспечением для анализа fast5. Это способствует лучшей интеграции в существующие рабочие процессы и стандартизации хранения данных. Новый формат обеспечивает лучшую компрессию данных, что позволяет сэкономить дисковое пространство при хранении больших объемов данных секвенирования.

Процесс перевода сигнала в формат fastq (файл с последовательностями нуклеиновых кислот и уровнем качества для каждого нуклеотидного остатка) называется декодированием (по-английски basecalling), для этого используют соответствующие программы, называемые декодерами. На основе амплитуды и формы сегментированных во времени сигналов декодеры определяют, какие нуклеотидные остатки соответствуют каждому сигналу. Более того, данные нанопорового секвенирования позволяют идентифицировать химические модификации нуклеотидных остатков, такие как 4mA (N⁴-метиладенин), 5mC (5-метилцитозин), 5hmC (5-гидроксиметилцитозин), 6mA (N⁶-метиладенин) и 8oxG (8-оксо-гуанин/8-оксо-2'-дезоксигуанозин) [24, 25]. Алгоритмы декодирования первичного сигнала в нанопоровом секвенировании прошли несколько этапов развития, начиная от простых статистических моделей до современных нейросетевых методов. На начальном этапе становления нанопорового секвенирования (2014 год) алгоритмы декодирования были основаны на скрытых марковских моделях (HMM, Hidden Markov Models) [26]. Так например первый декодер разработанный ONT – Metrichor [27], использовал скрытые марковские модели для предсказания последовательностей нуклеотидных остатков. Эти модели анализировали уровни сигнала, соответствующие проходящим через нанопору мотивам из 5 нуклеотидных остатков. Однако точность была относительно низкой, особенно для гомополимерных участков, где изменение сигнала было незначительным и сложным для интерпретации (если гомополимер более 5 при модели в 5 – дешифровка становится затруднена). В 2015 году компания ONT представила Nanonet – первую попытку внедрения нейросетевых подходов на основе рекуррентных нейронных сетей (RNN). Этот алгоритм был разработан для более точного декодирования сигналов, но оставался экспериментальным и не получил широкого распространения. Он использовал многослойные рекуррентные нейронные сети, которые анализировали временные зависимости в сигнале, что позволяло лучше учитывать контекст последовательности нуклеотидных остатков. Однако вычислительная сложность и низкая скорость работы ограничивали его использование. Постепенно с 2016 года разработчики начали более массово внедрять нейронные сети в новые версии декодеров, что позволило значительно повысить точность

распознавания сигналов. Так например Albacore (2016–2018) [28], использовал рекуррентные нейросети и метод CTC (Connectionist Temporal Classification) [29] для предсказания последовательностей. Этот алгоритм впервые применил нейросетевой подход для анализа временных рядов сигнала, что позволило лучше учитывать шум и структурные особенности данных [30]. Однако Albacore требовал значительных вычислительных ресурсов, что ограничивало его массовое использование. Еще один пример, Flappie (2018) [31] – экспериментальная программа ONT, основанная на двунаправленных рекуррентных нейросетях (BiRNN, Bidirectional Neural Networks) [32]. BiRNN позволяли учитывать контекстные зависимости сигнала в обоих направлениях, что улучшало точность декодирования. Однако из-за высокой вычислительной нагрузки и ограниченной поддержки со стороны разработчиков Flappie не получил широкого распространения.

Эти алгоритмы заложили основу для дальнейшего перехода к более эффективным сверточным нейросетям и гибридным моделям. В период с 2019 по 2022 год Oxford Nanopore полностью перешла на сверточные нейросети (CNN, Convolutional Neural Networks), которые оказались более устойчивыми к шуму и эффективными в обработке больших данных. Первым из них стал Guppy [28] – с 2018 по 2023 год главный декодер ONT, основанный на сверточных нейронных сетях. Guppy также поддерживал работу с GPU (Graphics Processing Unit) для ускорения декодирования. Экспериментальным декодером, в котором впервые применили модели трансформеров, стал Bonito [33]. Он позволил достичь высокой точности, но требовал значительных вычислительных ресурсов. С 2023 года по настоящее время алгоритмы главных декодеров основаны на использовании гибридных архитектур – сверточные нейронные сети + трансформер. Основным декодером на сегодняшний день, Dorado [34], отличается повышенной точностью на сложных участках генома и более высокой скоростью работы на GPU.

Кроме декодеров от ONT есть также множество программ от сторонних разработчиков. Для декодирования и анализа модификаций широко используют DeepMod 2 [35], DeepSignal [36], Nanodisco [37]. В некоторых случаях алгоритмы сторонних разработчиков показывают результат лучше, чем оригинальные программы. Так, например, было показано, что декодер DeepMod 2 определяет модификации 5mC лучше, чем оригинальный декодер Dorado [38].

Одним из новейших достижений в области декодирования редких модификаций ДНК является создание алгоритма для прямого определения 8oxG по данным нанопорового секвенирования [39]. Эта модификация возникает в результате окислительного повреждения гуанина. Разработан подход, основанный на использовании синтетических олигонуклеотидов, содержащих 8oxG в различных контекстах, что позволило обучить нейросетевую модель, способную с высокой специфичностью различать 8oxG от немодифицированного G. Алгоритм включает два этапа: модифицированную версию базового декодера Bonito, адаптированную для корректного считывания 8oxG, и последующую классификацию при помощи модели типа Remora [40], которая использует сырой сигнал и локальную последовательность для точного определения наличия модификации. Полученная модель позволяет одновременно определять как 8oxG, так и 5-mC на одной молекуле ДНК, открывая путь к изучению их взаимосвязей и роли в эпигенетической регуляции.

Таким образом, эволюция алгоритмов декодирования в нанопоровом секвенировании прошла путь от скрытых марковских моделей (HMM) и рекуррентных сетей (RNN) к современным сверточным нейросетям (CNN) и трансформерам, что позволило существенно повысить точность и скорость обработки данных.

1.2. Контроль качества данных секвенирования

Контроль качества данных секвенирования является критически важным вторым этапом после декодирования в анализе данных секвенирования. Он включает в себя проверку качества, фильтрацию и процессирование прочтений, полученных после декодирования.

Каждое прочтение в FASTQ-файле представлено четырьмя строками:

1. Идентификатор (начинается с @): @SEQ_ID
2. Последовательность нуклеотидных остатков: AGCTTAGCTAGCTACGATC
3. Разделитель (начинается с +, может повторять идентификатор или быть пустым): +
4. Качество считывания (ASCII-коды (от англ. American Standard Code for Information Interchange) символов представляют *Phred*-оценки [41]): !''*(((***+))

Каждый символ в четвертой строке кодирует *Phred*-оценку (*Q*-score, 1.1), определяющую вероятность ошибки:

$$Q = -10 \log_{10} P \quad (1.1)$$

где P – вероятность ошибки в определении нуклеотидного остатка.

Пример соответствия некоторых символов ASCII и значений *Phred* в таблице 1.

Таблица 1. Пример соответствия некоторых ASCII символов и *Phred*-оценок

Символ	<i>Phred</i>	Вероятность ошибки
!	0	1.0
#	2	0.63
+	10	0.1
?	30	0.001
I	40	0.0001

Кроме того, во время лигирования адаптерных последовательностей могут образовываться так называемые химерные прочтения [42], при которых одна адаптерная последовательность «сшивает» две разные молекулы нуклеиновой кислоты. В результате получается искусственное прочтение, содержащее фрагменты от разных исходных молекул. Поиск и разбиение химерных прочтений – это отдельная биоинформатическая задача, необходимая для повышения качества сборки и точности выравнивания. Программа Dorado basecaller автоматически удаляет адаптерные последовательности и разделяет химерные прочтения на этапе декодирования, если не указано иное поведение в параметрах запуска.

Для фильтрации прочтений по качеству, длине, а также для процессирования (удаления низкокачественных концов) можно использовать программы *chopper* [43] и *fastp* [44]. *Chopper* и *fastp* – это инструменты для предварительной обработки длинных прочтений, широко используемые в нанопоровом секвенировании. Они позволяют проводить фильтрацию, разделение химер и обрезку адаптерных последовательностей перед последующим анализом. Основные функции включают фильтрацию по длине (удаление слишком коротких, например, менее 100, или длинных прочтений), фильтрацию по качеству (отбрасывание прочтений с низкими *Phred*-оценками), а также

удаление первых или последних N нуклеотидных остатков, что помогает устранить артефакты секвенирования. Инструмент также позволяет разделять прочтения по заданным границам, обрезать адаптеры и баркоды, что особенно важно для корректной сборки и выравнивания данных. Праграмма *fastp* поддерживает различные режимы фильтрации по качеству. Один из них – фильтрация по среднему качеству, при которой прочтение удаляется, если его общее среднее *Phred*-качество ниже установленного порога. Более гибким для фильтрации по качеству является метод «скользящего окна» (*sliding window quality filtering*), где анализ ведется не по всей длине прочтения сразу, а с использованием скользящего окна фиксированного размера, который задается пользователем. В каждом таком окне вычисляется среднее значение *Phred*-оценки, и если оно падает ниже порогового уровня, прочтение либо обрезается до этого места, либо полностью удаляется, если оставшаяся длина становится слишком короткой. Это помогает бороться с деградацией качества к 3'-концу длинных прочтений, сохраняя при этом их наиболее информативные части. *Chopper* также может фильтровать прочтения по гомополимерам – удалять последовательности с длинными повторами одного нуклеотидного остатка, что помогает снизить количество ошибок, характерных для нанопорового секвенирования.

Для анализа качества полученных после фильтрации *fastq* файлов используют такие программы, как *FastQC* [45], *FastQscreen* [46] и *MultiQC* [47]. *FastQC* рассчитывает такие метрики, как распределение длин прочтений, распределение средних оценок качества (*Phred Score*) по позициям в прочтениях, наличие адаптерных последовательностей и некоторые другие. *FastQscreen* позволяет оценить контаминацию образца чужеродными молекулами ДНК, не относящимися к исследуемому образцу. *MultiQC* – это программа-агрегатор результатов выдачи различных программ для анализа данных секвенирования, она позволяет собирать выдачу разных программ в один удобный HTML-отчет.

1.3. Сборка контигов

Сборкой контигов называется объединение прочтений в более длинные последовательности, называемые контигами. Этот процесс требует использования специализированных программ, известных как сборщики (от английского *assembler*). Сборщики, алгоритмы которых используют графовые структуры, можно разделить на три категории. Методы *Overlap/Layout/Consensus (OLC)* используют граф перекрытий (Figure 1). Методы *de Bruijn Graph (DBG)* (Figure 2) работают с графами *k*-меров [48]. Жадные графовые алгоритмы (*greedy extension* [49]) могут использовать как *OLC*, так и *DBG* [50].

Граф – это абстрактная структура данных, состоящая из узлов и рёбер, соединяющих их. Если рёбра графа имеют направление, такой граф называется ориентированным. Визуально его можно представить как систему узлов, соединённых стрелками. Каждое ориентированное ребро представляет связь от одного узла к другому. Совокупность рёбер образует пути, следуя которым можно посещать вершины в определённом порядке.

Простой путь в графе – это путь, который не содержит повторяющихся узлов (каждый узел посещается не более одного раза). Простой путь не пересекает сам себя, а в некоторых случаях требуется, чтобы другие пути также не пересекали его. Узлы и рёбра могут иметь различные атрибуты и семантические значения, что важно при применении графов к задачам секвенирования.

Граф перекрытий представляет собой модель, в которой узлы соответствуют прочтениям, а рёбра – перекрытиям между ними. Прежде чем построить такой граф, необходимо вычислить все перекрытия прочтений. В реальном графе могут быть

дополнительные элементы и атрибуты, такие как 5'- и 3'-концы прочтений, ориентация прочтений (прямая и обратная комплементарная), длина прочтений, длина перекрытий и тип перекрытия (suffix-to-prefix или containment. Перекрытие типа suffix-to-prefix означает, что суффикс одного прочтения совпадает с префиксом другого, тогда как containment описывает ситуацию, когда одно прочтение полностью содержится внутри другого). Пути в графе соответствуют возможным контигам, и их можно перевести в последовательности (рис. 1).

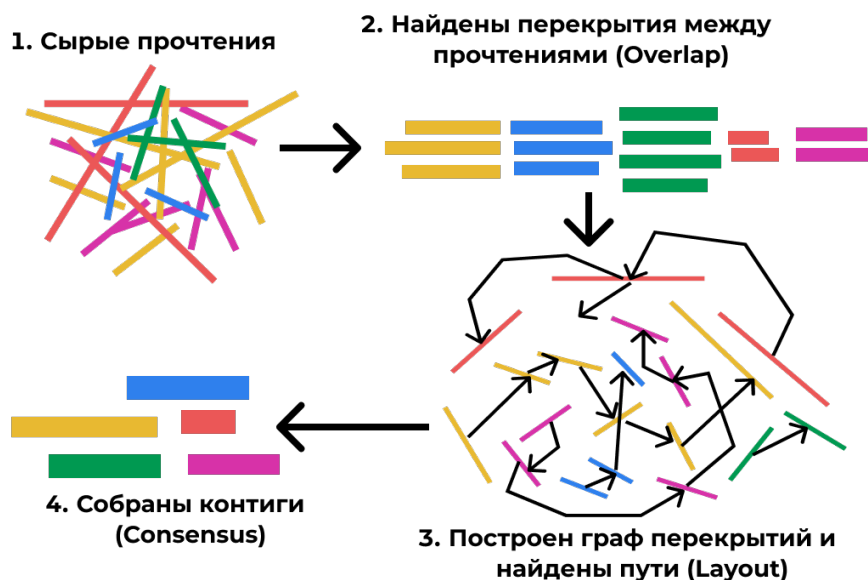


Рис. 1. Схематическое представление сборки генома с использованием графов перекрытий (OLC, Overlap-Layout-Consensus).

Для учета двухцепочечной природы ДНК применяются два подхода [51]:

1) Разделение узлов для концов прочтений, при котором путь должен выходить с противоположного конца, чем тот, через который он вошел.

2) Разделение рёбер для прямой и обратной цепи, при котором путь должен выходить на той же цепи, на которой он вошел.

Граф де Брёйна был изначально разработан для анализа строк конечного алфавита. В этом графе узлы представляют все возможные строки фиксированной длины (k -меры), а рёбра – идеальные перекрытия типа «suffix-to-prefix». Идеальные перекрытия (perfect overlaps) – это такие перекрытия между двумя строками (прочтениями или k -мерами), которые совпадают полностью без ошибок, то есть без пропусков, замен или вставок. Граф k -меров является частным случаем графа де Брёйна, где узлы представляют все k -меры, полученные из исходной последовательности, а рёбра представляют перекрытия $k-1$ оснований между последовательными k -мерами. Существуют два подхода к построению графа k -меров (рис. 2). В первом случае узлы соответствуют $(k-1)$ -мерным последовательностям, а рёбра – k -мерам, таким образом, каждое k -мерное слово создаёт одно ребро (Эйлеров граф де Брёйна). Во втором случае узлы представляют k -меры, а рёбра соединяют k -меры, имеющие общую $(k-1)$ -мерную часть (гамильтонов граф де Брёйна).

В любом случае, построенный граф содержит путь, соответствующий исходной последовательности. В узлах, где k -мер встречается несколько раз, пути в графе сходятся на себя, что отражает повторы в геноме.

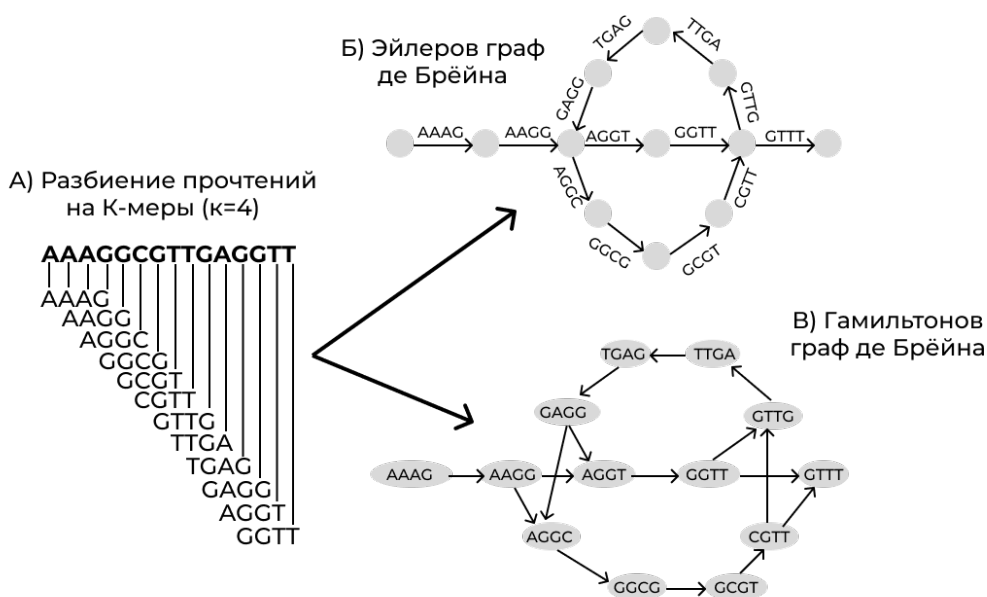


Рис. 2. Схематическое представление разбиения прочтения на k -меры и построения графов де Брёйна.

Для длинных прочтений, полученных с помощью нанопор, лучше всего подходит метод OLC-графов [52]. Метод графов де Брёйна в основном используется для сборки коротких прочтений, полученных на платформах секвенирования второго поколения. Так как в длинных прочтениях гораздо ниже точность секвенирования (на момент 2025 года), чем в коротких, разбиение на k -меры дает больше ошибок. В то же время OLC-граф более универсален, так как строится на основе реальных перекрытий прочтений, что позволяет учитывать ошибки секвенирования и создавать более точные сборки. Узлы в OLC-графе представляют собой полные прочтения, а рёбра соединяют их на основании их перекрытий, что делает его особенно эффективным для длинных прочтений с высокой вариабельностью.

Метод greedy extension находит ближайшие прочтения по схожести и последовательно объединяет их, но этот подход плохо масштабируется и склонен к накоплению ошибок. Он может работать на коротких прочтениях, но при сборке длинных прочтений он теряет точность и приводит к чрезмерной фрагментации сборки.

Для сборки длинных прочтений наиболее часто [53] используются программы Flye [54], Miniasm [55] и Canu [56].

1.3.1. Алгоритм сборки контигов с помощью графа повторов. Flye

Рассмотрим алгоритм работы наиболее популярного сборщика Flye. Flye использует уникальный метод представления повторов в геноме через граф повторов (repeat graph). Главная проблема сборки длинных чтений заключается в высоком уровне ошибок и сложной структуре повторяющихся элементов в геноме, что делает традиционные графовые подходы (например графы Де Брёйна) менее эффективными. Flye решает эту проблему, строя промежуточные структуры данных, которые позволяют постепенно уточнять геномную последовательность, исправлять ошибки и разделять повторяющиеся регионы.

Процесс сборки начинается с формирования структур, именуемых дизъюнктами (disjointigs). В отличие от большинства сборщиков, Flye не пытается сразу создать

точные контиги, а вместо этого строит последовательности, содержащие произвольные соединения участков генома. Эти дизъюнктивы получаются путём последовательного соединения длинных чтений, но без учёта их точной структуры и корректности. Этот этап необходим для того, чтобы быстро создать начальное представление возможной структуры генома, даже если оно содержит ошибки. После этого все дизъюнктивы сшиваются в одну длинную последовательность, где порядок сегментов может не соответствовать реальному геному.

На следующем этапе Flye строит график повторов (repeat plot) – двумерный график выравнивания этой последовательности на саму себя. Этот график позволяет выявить повторяющиеся области, так как они появляются в виде диагональных линий на диаграмме самовыравниваний. Используя этот график, программа строит начальный граф повторов (punctilious repeat graph), в котором схожие последовательности объединяются в узлы. Однако такой граф может оказаться слишком сложным и содержать большое количество ветвлений и коротких рёбер, что затрудняет дальнейший анализ. Чтобы устранить лишние сложности, Flye упрощает этот граф, удаляя короткие рёбра и агрегируя близкие по последовательности узлы, создавая финальный граф повторов (repeat graph).

В построенном графе повторов рёбра соответствуют повторяющимся сегментам генома, а вершины – уникальным регионам. Чтобы правильно реконструировать геном, Flye использует две ключевые стратегии разрешения повторов. Мостовые повторы (bridged repeats) – это те, которые полностью покрываются длинными чтениями, связывающими их начало и конец. В этом случае Flye может уверенно определить, какие копии повтора существуют, и разложить их на отдельные пути в графе. Немостовые повторы (unbridged repeats) – это повторы, которые не покрыты чтениями, пересекающими их полностью. Разделить такие повторы сложнее, так как нет прямых свидетельств их различий. Для их разрешения Flye использует небольшие вариации между копиями повтора (мутации, индельные различия) и пытается распределить чтения между разными вариантами, что позволяет сформировать несколько слегка отличающихся консенсусных последовательностей.

После разрешения повторов Flye извлекает из графа повторов пути, которые представляют собой окончательные контиги. Эти контиги более точны, чем первоначальные дизъюнктивы, так как в процессе их формирования были учтены ошибки и повторы. Благодаря этому Flye обеспечивает более качественные и непрерывные сборки по сравнению с другими программами, особенно в геномах с высокой долей повторов и сложными структурными вариациями.

1.4. Выравнивание прочтений на контиги

Длинные прочтения секвенаторов третьего поколения открывают новые возможности в геномных исследованиях благодаря своей большой длине (тысячи – сотни тысяч оснований), но их выравнивание к референсному геному представляет серьёзную вычислительную задачу. Главные сложности связаны с высокой частотой ошибок (сырые прочтения ONT могут иметь ошибки 1–2%, тогда как у Illumina-прочтений < 0.01%) и значительной длиной последовательностей. Классические методы для выравнивания коротких прочтений с этой задачей не справляются: высокие ошибки приводят к тому, что существенная доля длинных прочтений просто не выравнивается. Поэтому разработаны специализированные алгоритмы выравнивания длинных прочтений, сочетающие несколько парадигм: динамическое программирование для точного выравнивания, графовые представления для учета сложных вариантов и эвристики для ускорения работы. Примеры таких программ – Minimap2 [57], NGMLR [58] и GraphMap [59].

Наиболее распространённым инструментом для выравнивания длинных прочтений является Minimap2. Его высокая эффективность достигается за счёт применения индексации на основе минимизаторов, что позволяет значительно сократить память и время вычислений. Программа поддерживает широкий спектр задач: от выравнивания отдельных длинных прочтений и до попарного выравнивания целых геномов. Благодаря своей гибкости, точности и скорости работы Minimap2 является предпочтительным выбором во многих конвейерах.

Другой популярный инструмент, NGMLR, был специально разработан для выявления больших структурных вариаций. NGMLR включает в себя механизмы работы с ошибками секвенирования, основанные на вероятностной модели, что делает его особенно эффективным при выравнивании прочтений на участках со структурными перестройками (инверсии, крупные делеции, дупликации и транслокации). NGMLR обеспечивает высокую чувствительность и специфичность в задачах поиска сложных геномных событий. Сложные геномные события – это большие и нетривиальные изменения в структуре ДНК, которые сложно обнаружить стандартными методами. К таким событиям относятся инверсии (разворот участка ДНК в обратную сторону), крупные делеции (удаление длинных фрагментов), дупликации (удвоение участков), транслокации (перемещение фрагмента в другое место, в том числе на другую хромосому), а также комбинированные варианты, например, инверсия с дупликацией или перестройка с несколькими разрывами. Эти изменения нарушают обычный порядок нуклеотидов и требуют специальных алгоритмов для корректного выявления, особенно при работе с длинными прочтениями.

Инструмент GraphMap основан на графовом подходе к выравниванию длинных прочтений. Его алгоритм строит ориентированный ациклический граф, в котором вершины представляют потенциальные совпадения между участками прочтений и референсной последовательности, а рёбра соответствуют возможным путям выравнивания. Такой подход позволяет учитывать множественные пути, включая вариации, ошибки и пропуски, которые часто встречаются в длинных прочтениях. После построения графа GraphMap применяет модифицированную версию алгоритма динамического программирования для нахождения наиболее правдоподобного пути, соответствующего точному выравниванию прочтения. Это обеспечивает высокую чувствительность даже в случае прочтений с высоким уровнем ошибок. GraphMap особенно хорошо работает на участках генома с высокой вариабельностью и может использоваться для задач, требующих точного локального выравнивания, таких как выявление структурных вариантов или сборка гаплотипов.

Таким образом, разнообразие подходов к выравниванию длинных прочтений позволяет эффективно решать разные биоинформатические задачи. Выбор инструмента во многом зависит от специфики решаемой задачи, характеристик исходных данных и вычислительных ресурсов исследователя.

Рассмотрим алгоритм наиболее популярной программы для выравнивания длинных прочтений – Minimap2.

1.4.1. Выравнивание с Minimap2

Алгоритм Minimap2 [57] следует общей парадигме «seed-chain-align» (затравка-цепочка-выравнивание), традиционной для программ построения выравниваний. На первом этапе программа извлекает минимизаторы. Каждый минимизатор представляется в виде хеш-значения, то есть числового кода, полученного с помощью хеш-функции. Minimap2 сохраняет эти хеши в хеш-таблице, где ключом служит хеш минимизатора, а значением – список координат всех его вхождений в референсе.

На этапе поиска, для каждой входной последовательности Minimap2 также извлекает минимизаторы и ищет точные совпадения с минимизаторами из референса, формируя якоря – пары координат, указывающие на локальные совпадения между запросом и референсом.

Затем из полученных якорей Minimap2 строит цепочки (chains), которые представляют собой наборы коллинеарных якорей, то есть упорядоченных по координатам на обоих участках. Оценка качества цепочки вычисляется с помощью динамического программирования: вводится функция $f(i)$ (1.2), отражающая наилучший балл (score) цепочки, заканчивающейся на i -м якоре, с учётом расстояний между якорями и штрафов за разрывы.

$$f(i) = \max \left\{ \max_{j \geq 1} \{f(j) + \alpha(j, i) - \beta(j, i)\}, w_i \right\} \quad (1.2)$$

где:

- $f(i)$ – лучший «балл» цепочки, заканчивающейся на якоре i ;
- $\alpha(j, i)$ – число совпадающих нуклеотидных остатков между якорями j и i ;
- $\beta(j, i)$ – стоимость разрыва между якорями j и i ;
- w_i – длина затравки (seed length) для якоря i , используемая как начальное значение, если у i нет подходящих предшественников.

Для повышения производительности реализована эвристика: перебор кандидатов прекращается после фиксированного числа (обычно $h = 50$) итераций, если не удаётся улучшить результат. Это снижает вычислительную сложность построения цепочек с $O(N^2)$ до $O(hN)$, где N – число якорей. После построения всех возможных цепочек, Minimap2 определяет первичные цепочки (primary chains), исключая перекрывающиеся более чем на 50 % на запросе (то есть на прочтении, который выравнивается на референс), и оценивает качество выравнивания (mapping quality, $MAPQ$, 1.3) на основе отношения баллов лучшей и следующей по качеству цепочки по следующей формуле:

$$MAPQ = 40 \cdot \left(1 - \frac{f_2}{f_1}\right) \cdot \min\left(1, \frac{m}{10}\right) \cdot \ln f_1 \quad (1.3)$$

где:

- m – количество якорей в первичной цепочке;
- f_1 – балл (вес) первичной цепочки;
- $f_2 \leq f_1$ – балл лучшей вторичной цепочки, перекрывающейся с первичной.

Интуитивно, чем длиннее цепочка и чем слабее её ближайший конкурент (вторичная цепочка), тем выше значение $MAPQ$.

На следующем этапе Minimap2 выполняет глобальное выравнивание между якорями в цепочке. Для этого применяется динамическое программирование с двухчастной аффинной функцией штрафов за разрывы (2-piece affine gap cost), позволяющей различать короткие и длинные вставки/делеции и более точно выравнивать длинные прочтения с инделами. Этот этап реализован с использованием формулы Судзуки – Касахары [60], позволяющей Minimap2 значительно ускорять выравнивание с помощью параллелизации.

При построении глобального выравнивания алгоритм Minimap2 может ошибочно выравнивать неродственные участки между двумя якорями. Чтобы избежать этого артефакта, используется эвристика *Z-drop*, которая анализирует накопленный балл вдоль

пути выравнивания и обрывает выравнивание, если он резко падает в диагональном направлении.

Пусть $S(i, j)$ – балл выравнивания (см. 1.4), соответствующий ячейке (i, j) в матрице динамического программирования. Выравнивание обрывается, если существуют (i', j') и (i, j) , такие что $i' < i$ и $j' < j$, и выполнено следующее неравенство:

$$S(i', j') - S(i, j) > Z + e \cdot |(i - i') - (j - j')| \quad (1.4)$$

где:

- e – стоимость расширения разрыва (gap extension cost),
- Z – произвольный порог (threshold), задаваемый пользователем. Он определяет, насколько сильно может «упасть» балл выравнивания, прежде чем процесс будет остановлен.

Данный подход впервые был реализован BWA-MEM [61] и схож с эвристикой X-drop, применяемой в BLAST [62]. Однако, в отличие от X-drop, Z-drop не прерывает выравнивание при наличии одного длинного разрыва.

Если Minimap2 прерывает глобальное выравнивание между двумя якорями, он выполняет дополнительное локальное выравнивание между соответствующими подпоследовательностями, при этом одна из них берётся в обратной комплементарной форме. Этот шаг позволяет выявить короткие инверсии, которые могли быть пропущены на этапе построения цепочек.

Также предусмотрена фильтрация ошибочных якорей, которые могут приводить к локально неверному выравниванию. Например, отбрасываются якоря, которые вызывают одновременно вставку и делецию более 10 пар оснований между двумя соседними точками. Это дополнительно повышает надёжность финального выравнивания.

1.5. Коррекция контигов

Коррекция контигов (или полировка, от английского polishing) является завершающим этапом в процессе сборки генома (или метагенома), направленным на улучшение качества собранных последовательностей, а именно – коррекцию ошибок. После начальной сборки контигов, полученные последовательности могут содержать ошибки, такие как вставки, делеции и замены, которые возникли из-за ошибок секвенирования или алгоритмов сборки. Коррекция помогает минимизировать эти ошибки и получить последовательности максимально приближенные к реальной последовательности генома.

Основные этапы коррекции контигов:

1. Выравнивание прочтений на контиги: На этом этапе сырые прочтения повторно выравниваются на собранные контиги с помощью таких прокрам как Minimap2. Этот процесс позволяет установить консенсус, а также идентифицировать и локализовать ошибки в собранных последовательностях.
2. Коррекция ошибок: Алгоритмы коррекции, такие как Pilon [63], Racon [64], или Medaka [65], применяют различные методы для корректировки контигов. Pilon, например, использует короткие прочтения для точного исправления, тогда как Racon и Medaka работают с длинными прочтениями и используют подходы машинного обучения.
3. Оценка качества: После коррекции необходимо оценить качество полученных последовательностей. Для этого могут использоваться инструменты, такие как

Mercury [66], Inspector [67], KAT [68], которые предоставляют подробные отчеты о качестве сборки, включая такие показатели, как длина контигов, количество ошибок и степень покрытия.

Программа для коррекции контигов, которую рекомендует компания ONT, Medaka, реализует стратегию с опорой на нейросетевую модель, обученную на больших объёмах парных данных: необработанные прочтения и соответствующие им корректные последовательности. На входе принимаются предварительно собранный референс (например, контиг после сборки), выравнивания исходных прочтений в формате BAM, а также модель нейронной сети, подходящая для используемой версии нанопоры и набора реактивов. Выравнивания прочтений должны быть выполнены с высокой точностью, что обычно достигается с помощью инструмента Minimap2 с параметрами по умолчанию, оптимизированными для длинных прочтений. После инициализации Medaka разбивает собранную последовательность на перекрывающиеся участки фиксированной длины (обычно 1–2 Кб), с небольшим нахлёстом между окнами. Эти окна подаются на вход нейросетевой модели, которая по множеству выравненных прочтений, покрывающих каждый участок, строит вероятностную оценку последовательности нуклеотидных остатков. Архитектура модели, реализованной в Medaka, представляет собой вариацию сверточной нейронной сети (CNN), обученной распознавать ошибочные вставки, делеции и замены, типичные для данных нанопорового секвенирования. На этом этапе создаётся так называемая pileup-модель, которая агрегирует сигналы от выравненных прочтений. Нейросеть предсказывает наиболее вероятную «истинную» последовательность для каждого окна, включая все возможные локальные мутации по сравнению с изначальной сборкой. Результаты обработки каждого окна затем собираются в итоговую, скорректированную последовательность. Благодаря перекрытиям между окнами обеспечивается непрерывность и согласованность результата, при этом особые алгоритмы сглаживания устраняют артефакты, возникающие на границах окон. Выходной результат – это FASTA-файл с более точной последовательностью нуклеотидных остатков.

1.6. Анализ качества сборки *de novo*

Для оценки качества собранных *de novo* (не имеющих референса или ранее не известных) геномов, полученных из длинных прочтений, используются программы, работающие без использования референсного генома. Одной из первых таких программ является KAT [68], которая применяет анализ k -мерных спектров для выявления ошибок сборки и позволяет визуально оценивать полноту и корректность полученных контигов. Дальнейшим развитием этого подхода стала программа Mercury [66], добавившая функционал для определения полноты и точности фазирования гаплотипов в случае трио-анализов. Ещё одной широко используемой программой является Inspector [67], которая детектирует структурные и мелкомасштабные ошибки в сборках, корректирует их путём локальной пересборки проблемных участков с использованием исходных длинных прочтений, а также рассчитывает оценки качества. Эти инструменты предоставляют исследователям эффективные и точные средства для детальной оценки и улучшения качества геномныхборок.

1.6.1. Оценка качества сборки генома *de novo* с помощью Inspector

Inspector представляет собой программное обеспечение для комплексной оценки качестваборок геномов, полученных методом *de novo* с использованием длинных прочтений, которое работает без использования референсного генома, схематически этот подход продемонстрирован на рисунке 3. Основное преимущество Inspector заключается

в его способности детектировать структурные ошибки (например, коллапсы, расширения, гаплотипные переключения и инверсии) и мелкомасштабные ошибки (замены оснований, небольшие вставки и делеции менее 50 bp), различая их от истинных генетических вариантов. Для этого программа анализирует выравнивания прочтений на контиги с использованием Minimap2, после чего статистически фильтрует и выделяет достоверные ошибки, опираясь на частоту прочтений, подтверждающих ошибку, и локальное покрытие. Важной особенностью является встроенный модуль коррекции ошибок.

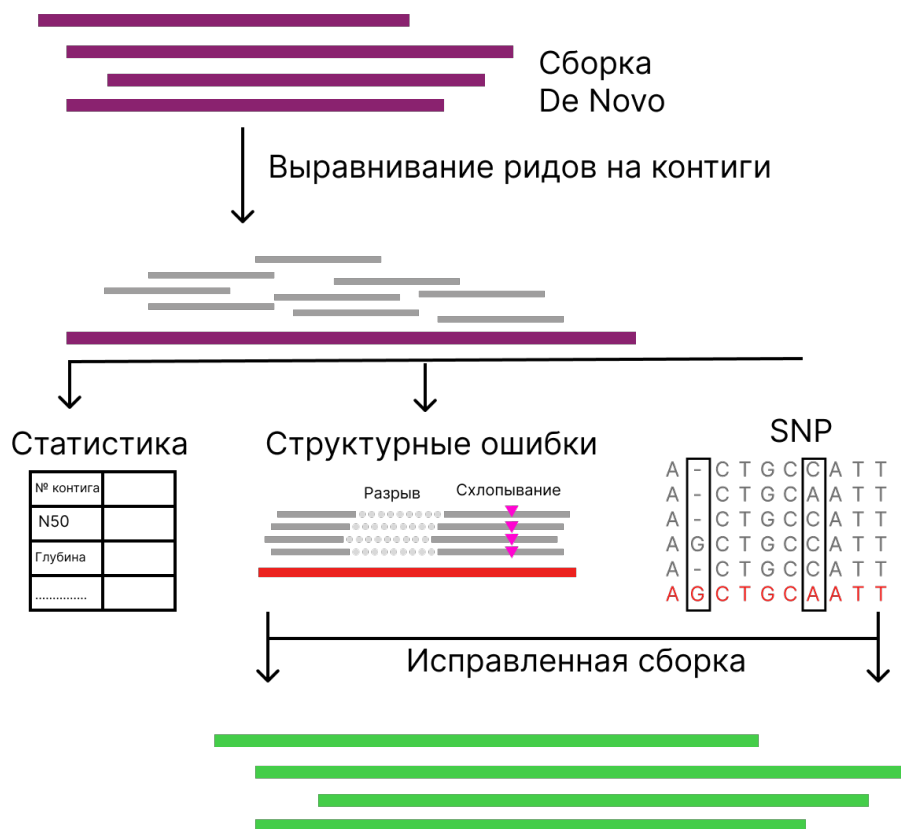


Рис. 3. Алгоритм Inspector для оценки качества сборки генома *de novo*; лиловым цветом показаны контиги на входе программы, а зелёным – исправленные контиги на выходе.

Inspector вычисляет сводные параметры статистического анализа, отражающие общую точность и качество сборки генома, среди которых ключевым является показатель *QV* (Quality Value). Он который рассчитывается на основе количества и масштаба выявленных ошибок (структурных и мелкомасштабных), нормированных на общую длину полученной сборки. Формула вычисления *QV* учитывает суммарное количество оснований, затронутых всеми типами ошибок, и преобразует это число в стандартную логарифмическую шкалу (*Phred*), широко используемую в биоинформатике для оценки качества последовательностей.

1.6.2. Оценка качества сборки генома *de novo* с помощью Merquy

Алгоритм работы программы Merquy основан на анализе *k*-меров и включает в себя несколько последовательных этапов. Сначала производится подсчёт *k*-меров в прочтениях и отдельно в последовательностях сборки. Затем программа сравнивает

наборы k -меров из прочтений и сборки, что позволяет выявлять и характеризовать ошибки сборки, такие как ложные дубликации или пропущенные фрагменты. Важной частью алгоритма является генерация копийных спектров (copy number spectrum), отражающих количество появлений каждого k -мера в прочтениях и его соответствие количеству в сборке. На основе этого спектра можно визуальнo и статистически оценить полноту и корректность сборки. Merqury также способна идентифицировать гаплотип-специфичные k -меры (hap-mers) с использованием данных от родительских геномов (если доступен трио-анализ), что позволяет оценить качество фазирования гаплотипов и выявить ошибки переключения фаз (phase switch errors). На следующем этапе алгоритм вычисляет показатель качества консенсуса (QV score) – логарифмическую оценку вероятности ошибки на каждое основание сборки, основываясь на доле уникальных для сборки k -меров, отсутствующих в прочтениях. Наконец, Merqury предоставляет различные графические отчёты, включая k -мерные спектры и визуализацию гаплотипных блоков, что облегчает интерпретацию и дальнейший анализ результатов. Благодаря такому подходу Merqury позволяет комплексно и точно оценивать качество геномных сборок без необходимости использования референсного генома.

При использовании k -меров существуют ограничения на их размер (значение k). Эти ограничения зависят от нескольких факторов. Во-первых, длина прочтений: значение k не может быть больше длины прочтения, иначе k -меры невозможно получить. Обычно k выбирается значительно меньше длины прочтений, например, 31. Во-вторых, структура генома: если k слишком маленькое, в данных будет много повторяющихся k -меров, что усложняет анализ. Если k слишком большое, возрастает вероятность, что даже близкие последовательности не будут совпадать, особенно при наличии ошибок в прочтениях. В-третьих, технические ограничения: при увеличении k экспоненциально растёт число возможных k -меров (4 в степени k), что требует больше памяти и вычислительных ресурсов. Например, при $k = 21$ возможны около 4 триллионов разных k -меров, а при $k = 31$ – уже около 4.6 квинтиллионов. Кроме того, программы, включая Merqury, имеют встроенные ограничения на максимальное значение k , в зависимости от формата хранения и архитектуры системы. Таким образом, выбор значения k – это компромисс между точностью, чувствительностью, вычислительными затратами и ограничениями конкретного инструмента.

1.7. Таксономическая классификация в метагеномном секвенировании

Метагеном – это совокупность генетического материала, извлечённого напрямую из образца, содержащего множество организмов, без необходимости выделения и культивирования отдельных видов. Метагеном охватывает все гены всех сообществ, присутствующих в исследуемом образце (например, почве, воде, кишечнике), и служит основой для изучения состава, функций и взаимодействий микробиоты в её естественной среде обитания.

Таксономическая классификация – это процесс определения принадлежности генетических последовательностей к определенным таксонам, что является ключевым аспектом в метагеномном эксперименте. Этот процесс может проводиться двумя способами – классификация прочтений и классификация контигов.

Классификация прочтений. В методе классификации прочтений каждая отдельная последовательность анализируется независимо для определения его таксономической принадлежности. Этот подход часто использует базы данных известных геномов или таксономических маркеров. Программы сопоставляют прочтения с этими базами данных с помощью алгоритмов, похожих на поиск соответствий в генетических последовательностях. Основным преимуществом этого метода является его скорость, так

как обработка каждого прочтения происходит отдельно и не требует предварительной сборки генома и полировки. Однако такой метод в случае длинных прочтений, содержащих множество ошибок, работает плохо [69].

Классификация контигов. Скорректированные контиги помимо большей длины также имеют лучшую по сравнению с прочтениями точность. В этом подходе сначала проводится сборка генома или метагенома, в результате чего образуются контиги, которые затем корректируются и классифицируются по таксонам.

Существует несколько подходов к таксономической классификации последовательностей. Методы таксономической классификации, основанные на использовании k -меров (небольших последовательностей из k нуклеотидных остатков), обеспечивают эффективный и быстрый способ анализа и классификации генетических последовательностей. Вот несколько примеров популярных инструментов и методов, которые используют k -меры для таксономической классификации:

1. Kraken и Kraken2 [70, 71]: Это один из наиболее известных инструментов для таксономической классификации, который использует систему, основанную на k -мерах. Kraken строит базу данных k -меров из известных геномов, а затем сравнивает k -меры из исследуемых последовательностей (прочтений) с этой базой для определения таксономической принадлежности. Преимущество Kraken заключается в его высокой скорости и точности благодаря специфическому подходу к индексации и поиску k -меров. Kraken2 – улучшенная версия Kraken, которая использует меньше памяти и работает быстрее благодаря более эффективному методу индексирования. Kraken2 также способен работать с k -мерами различной длины, что позволяет ему быть более гибким в различных исследовательских сценариях.
2. Centrifuge [72]: Этот инструмент также основан на k -мерах и предназначен для классификации генетических последовательностей с высокой скоростью. Centrifuge использует компактный индекс, который позволяет ему быстро определить, какие k -меры присутствуют в исследуемой последовательности и к каким таксонам они могут принадлежать.
3. Kaiju [73]: Kaiju является еще одним инструментом для классификации, который использует k -меры для анализа белковых последовательностей. В отличие от других инструментов, Kaiju преобразует прочтения ДНК в белковые последовательности и использует k -меры для сопоставления с базой данных белковых последовательностей.

Методы таксономической классификации, основанные на выравнивании последовательностей, предполагают сравнение неизвестных последовательностей с известными референсными последовательностями в базах данных. Эти методы могут предоставить более точную классификацию за счёт детального анализа геномных сходств и различий. Ниже перечислены некоторые из основных инструментов для таксономической классификации, основанные на выравнивании:

1. BLAST (Basic Local Alignment Search Tool) [74]: Это один из самых известных инструментов в биоинформатике. BLAST позволяет пользователям сравнивать интересующие их последовательности с базой данных последовательностей, выявляя регионы локального сходства. Эти сравнения могут использоваться для таксономической классификации, определяя, к каким организмам могут принадлежать интересующие последовательности.

2. MetaPhlAn 4 (Metagenomic Phylogenetic Analysis) [75]: Это четвёртая версия популярного инструмента, разработанного для метагеномной таксономической классификации микробных сообществ с использованием уникальных маркерных генов. MetaPhlAn использует базу данных, содержащую миллионы уникальных маркерных генов, ассоциированных с более чем 20000 таксономическими единицами. Инструмент анализирует метагеномные данные, чтобы определить состав микробиома, вычисляя относительные обилия различных микробов. MetaPhlAn 4 обеспечивает быстрое и точное выравнивание, позволяя пользователям быстро анализировать и интерпретировать сложные метагеномные данные.
3. DIAMOND [76]: Это высокопроизводительный инструмент для выравнивания белковых последовательностей, который является альтернативой BLAST, работающей на порядки быстрее. DIAMOND выполняет выравнивание аминокислотных последовательностей белков (или транслированных в разных рамках считывания последовательностей ДНК) против большой белковой базы данных, такой как NR от NCBI, с высокой скоростью и точностью. Это делает DIAMOND особенно подходящим для метагеномных исследований, где необходимо быстро анализировать большие объемы данных.
4. MEGAN (MEtaGenome ANalyzer) [77]: MEGAN анализирует данные, полученные после выравнивания с помощью таких инструментов как BLAST, для таксономической классификации на основе последовательностей. MEGAN использует данные выравнивания для создания дерева, отображающего таксономические отношения между различными последовательностями, где точкой ветвления является наименьший общий предок (Lowest Common Ancestor, LCA) всех таксонов, к которым могли бы быть отнесены выравненные последовательности.

1.7.1. Таксономическая классификация с анализом k -меров. Kraken2

Алгоритм классификации последовательностей

Для классификации ДНК-последовательности S выполняется сбор всех k -меров из данной последовательности в набор, обозначаемый как $K(S)$. Каждый k -мер из $K(S)$ сопоставляется, согласно алгоритму, описанному ниже, с таксоном LCA (наименьший общий предок) всех геномов, содержащих этот k -мер. Таксоны LCA и их предки в таксономическом дереве формируют классификационное дерево – поддереву, используемое для классификации последовательности S .

Каждому узлу классификационного дерева присваивается вес, равный количеству k -меров из $K(S)$, которые сопоставлены с таксоном, связанным с этим узлом. Для каждого пути от корня к листу (*Root-to-Leaf Path*, RTL) в классификационном дереве выполняется расчет оценки путем суммирования всех весов узлов вдоль пути. Путь RTL с максимальной оценкой определяется как классификационный путь, и последовательность S получает метку, соответствующую листу этого пути. Если таких путей несколько, метка присваивается по таксону LCA всех листьев этих путей.

Этот алгоритм, проиллюстрированный на рисунке 4, позволяет программе Kraken рассматривать каждый k -мер последовательности как отдельное доказательство. При правильно выбранной длине k большинство k -меров однозначно сопоставляются с одним видом, что значительно упрощает процесс классификации. Последовательности, в которых ни один из k -меров из $K(S)$ не обнаружен в геномах базы данных, остаются неклассифицированными.

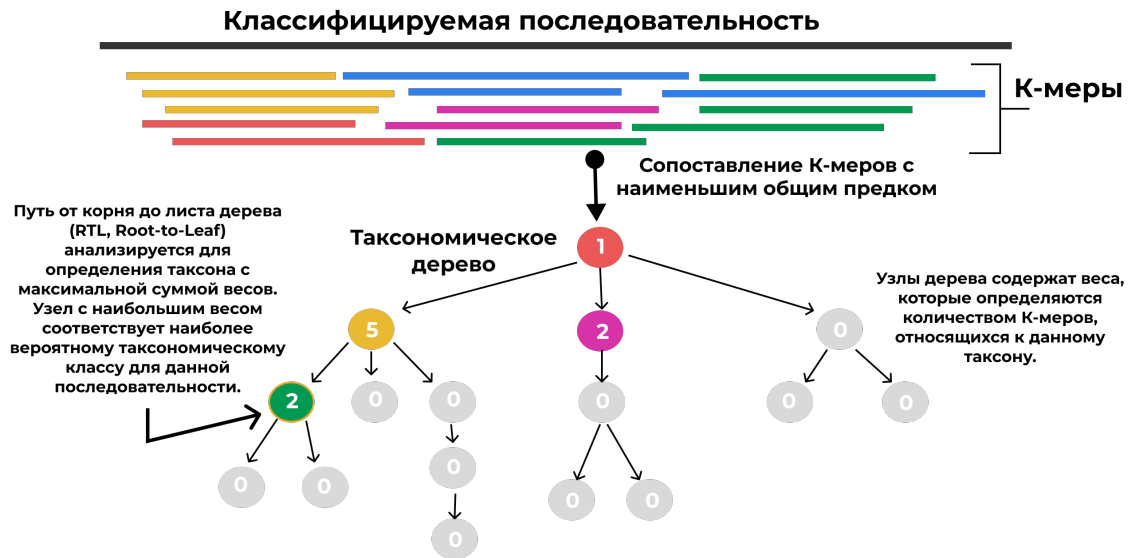


Рис. 4. Описание алгоритма работы таксономического классификатора kraken.

Использование оценки пути RTL в классификационном дереве необходимо в связи с неизбежными различиями между классифицируемыми последовательностями. Эти различия могут приводить к тому, что даже при больших значениях k k -меры, содержащиеся в библиотеке, будут ассоциированы с видами, далекими от истинного источника последовательности. Оценка различных путей RTL в классификационном дереве позволяет компенсировать эти различия и корректно классифицировать последовательности, даже если небольшая часть k -меров указывает на ошибочную таксономическую принадлежность.

Создание базы данных

Эффективная реализация алгоритма классификации Kraken требует, чтобы сопоставление k -меров с таксонами выполнялось путем запроса к предварительно рассчитанной базе данных. Kraken создает эту базу данных в ходе многоэтапного процесса, начиная с выбора библиотеки геномных последовательностей. Kraken по умолчанию включает в себя стандартную библиотеку, состоящую из геномов микроорганизмов в базе данных RefSeq NCBI (www.ncbi.nlm.nih.gov/refseq), но база данных может быть собрана из любых последовательностей по желанию пользователя [78].

Для работы с библиотекой используется программа Jellyfish [79], которая представляет собой многопоточный инструмент для подсчета k -меров. Jellyfish генерирует базу данных, содержащую все уникальные k -меры длиной 31 символ (параметр по умолчанию, пользователь может задать свое значение) из выбранной библиотеки. На данном этапе каждый k -мер сохраняется в базе данных вместе с его числом вхождений, но эти данные будут переработаны на следующем этапе.

После создания базы данных с использованием Jellyfish, Kraken использует выделенные ранее 4-байтовые ячейки, которые хранили информацию о частотах k -меров, для записи таксономических идентификаторов (Taxonomic ID) значений LCA. Это позволяет эффективно использовать память и минимизировать объем хранимых данных.

Каждая геномная последовательность из библиотеки обрабатывается поочередно. Для каждой последовательности определяется связанный с ней таксон. Затем для всех k -меров, содержащихся в этой последовательности, устанавливается значение LCA, соответствующее текущему таксону.

Если k -мер из текущей последовательности уже имеет установленное значение LCA (например, этот k -мер был встречен ранее в другой последовательности), то вычисляется новое значение LCA, объединяющее ранее сохраненное значение и текущий таксон. Это обеспечивает правильное назначение k -меров к их наименьшим общим предкам.

Информация о таксонах, необходимая для сопоставления и вычисления значений LCA, извлекается из таксономической базы данных NCBI www.ncbi.nlm.nih.gov/Taxonomy. Это гарантирует высокую точность и согласованность при работе с таксономическими уровнями (род, семейство, класс и т.д.). После завершения всех этапов создается оптимизированная база данных, где каждому уникальному k -меру присвоен таксономический идентификатор, соответствующий его наименьшему общему предку. Эта база данных используется алгоритмом Kraken для быстрого сопоставления k -меров из исследуемых последовательностей с таксонами и классификации данных. Такой подход обеспечивает высокую скорость анализа и минимизацию вычислительных ресурсов.

Выбор длины k -меров

Длина k -меров в 31 символ в алгоритме Kraken выбрана не случайно, она обеспечивает высокую вероятность того, что k -мер будет уникальным для определённого генома или вида.

- Слишком короткие k -меры (например, 10–20 символов) могут встречаться в последовательностях многих организмов из-за сходства геномов (консервативных регионов). Это приведёт к снижению точности классификации, так как несколько таксонов могут быть связаны с одним и тем же k -мером.
- Слишком длинные k -меры (например, больше 50 символов) могут быть уникальными, но при этом более чувствительны к ошибкам секвенирования. Даже одна ошибка (например, вставка, делеция или замена нуклеотидного остатка) может сделать длинный k -мер непригодным для сопоставления.

Длина k -меров в 31 символ стала своего рода стандартом в инструментах для анализа последовательностей [80] [81]. Это позволяет разрабатывать универсальные базы данных и сопоставимые конвейеры, что упрощает интеграцию разных программ и инструментов.

Структура базы данных и алгоритм поиска в Kraken

Использование минимизаторов. Kraken оптимизирует процесс поиска k -меров в базе данных с помощью концепции минимизаторов (рис. 6). Минимизатор – это наименьший (в лексикографическом или числовом порядке) k -мер внутри окна фиксированной длины w , скользящего по последовательности. Он служит компактным представлением окна и используется для ускорения поиска и сравнения последовательностей. В отличие от хранения всех k -меров, хранение только минимизаторов позволяет существенно сократить объём индексируемых данных и ускорить вычисления.

1. Каноническое представление последовательности. Каноническое представление ДНК-последовательности S определяется как лексикографически меньшая

из самой последовательности S и её обратного комплемента. Это позволяет стандартным образом представлять последовательности независимо от их ориентации.

2. Определение минимизатора. Для определения минимизатора длиной M у k -мера длиной k , из всех M -меров внутри данного k -мера выбирается лексикографически минимальный M -мер (в его каноническом представлении). Таким образом, минимизатор характеризует k -мер.
3. Группировка k -меров в базе данных. Все k -меры, имеющие один и тот же минимизатор, хранятся последовательно и упорядочиваются в лексикографическом порядке их канонических представлений. Это позволяет ускорить процесс поиска, так как для каждого минимизатора можно сразу обратиться к соответствующему участку базы данных.

Алгоритм поиска

1. Поиск минимизатора. Для заданного k -мера R определяется минимизатор M_R .
2. Определение диапазона поиска. С помощью индекса базы данных определяется диапазон, где хранятся все k -меры с минимизатором M_R . Индекс хранит позиции начала каждой группы минимизаторов в базе данных.
3. Бинарный поиск. После определения диапазона выполняется бинарный поиск среди k -меров внутри этого диапазона для нахождения R .
4. Оптимизация последовательных запросов. Если минимизатор остаётся неизменным между двумя соседними запросами, то диапазон поиска также остаётся тем же. Это позволяет повторно использовать данные из кэша процессора, вместо обращения к оперативной памяти (от англ. Random Access Memory, RAM), что ускоряет выполнение запросов.

Распределение минимизаторов. Каноническое лексикографическое упорядочивание M -меров может создавать смещения в распределении минимизаторов, особенно для низкосложных M -меров (например, состоящих из повторов). Это приводит к образованию крупных диапазонов поиска, замедляющих процесс.

Чтобы устранить этот эффект, в Kraken применяется операция исключающего ИЛИ (XOR), которая изменяет половину битов канонического представления M -мера перед его лексикографическим сравнением. Это создаёт равномерное распределение минимизаторов.

Использование предыдущего диапазона. Если для нового k -мера R минимизатор совпадает с минимизатором предыдущего k -мера, то сначала проверяется предыдущий диапазон поиска. Если k -мер находится, поиск завершается. Если его нет, то дальнейший поиск не производится, так как k -мер гарантированно отсутствует в этом диапазоне. Только если минимизатор изменился, обновляется диапазон поиска, и процесс продолжается.

Минимизаторы в Kraken

- Для k -мера длиной $k = 31$ по умолчанию используется минимизатор длиной $M = 15$.

- Пользователь может изменить длину минимизатора. Например, для уменьшенной версии базы данных MiniKraken использовались минимизаторы длиной $M = 13$, чтобы общий размер базы данных не превышал 4 Гб.

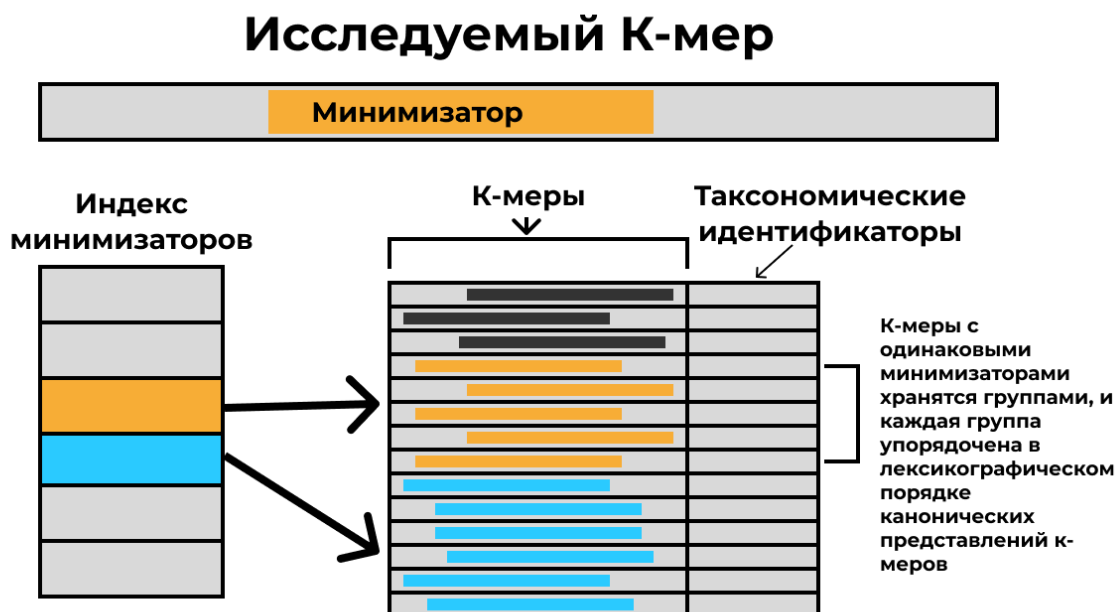


Рис. 5. Структура базы данных Kraken и процесс поиска k -меров с использованием минимизаторов.

1.7.2. Таксономическая классификация и количественный анализ в штрихкодировании 16S с Emu

Штрихкодирование 16S - это метод молекулярной идентификации видов в сложных биологических смесях на основе амплификации и секвенирования стандартизированных участков ДНК (баркодов), характерных для определённых таксонов. Чаще всего используется для анализа биоразнообразия в образцах окружающей среды (почва, вода, кишечное содержимое и др.), без необходимости культивирования организмов. В штрихкодировании 16S применяются универсальные праймеры, направленные на консервативные генные участки (например, 16S рРНК у прокариот, внутренние транскрибируемые спейсеры ITS1–ITS2 рРНК у грибов или фрагмент гена цитохром-оксидазы I (COI) митохондриальной ДНК у животных), что позволяет определить таксономический состав сообщества на основе сравнений с референсными базами данных.

Алгоритм Emu [82] основан на методе максимизации ожидания (ЕМ, expectation maximization) и применяется для оценки долей таксонов. Рассмотрим его подробно.

1. Вычисление вероятностей ошибок в выравнивании, $P(c)$

Софтклип (soft-clipped mutation) – это разновидность структурного варианта, выявляемая при анализе выравнивания прочтений к референсному геному, при которой часть прочтения (обычно с одного из концов) не выравнивается, но при этом сохраняется в данных. Такие участки могут указывать на наличие мутаций, не представленных в референсе, включая инсерции, делеции, транслокации, а также точки разрыва в случае

химерных или вирусных вставок. Анализ софтлип-областей особенно важен в задачах выявления новых или редких структурных вариаций, а также при работе с длинными прочтениями.

Вычисляются на основе частот наблюдаемых ошибок в первичных выравниваниях:

$$P(c) = \frac{n_c}{\sum_{c \in C} n_c}, \quad (1.5)$$

где C – множество типов ошибок (например, несовпадение X , инсерция I , делеция D , софтлип SJ), а n_c – количество наблюдений для ошибки c .

2. Инициализация вектора композиции, $F(t)$

Композиция таксонов $F(t)$ инициализируется равномерно для всех таксонов t в множестве T :

$$F(t) = \frac{1}{|T|}, \quad \forall t \in T. \quad (1.6)$$

3. Вычисление вероятностей выравнивания

Для каждого прочтения r и таксона t вычисляется вероятность выравнивания $P(r | t)$, которая определяется как максимальная вероятность между r и t :

$$P(r | t) = \max_{c \in C} \prod_{c \in C} P(c)^{\hat{n}_c(r,t)}, \quad (1.7)$$

где $\hat{n}_c(r, t)$ – количество операций типа c , связанных с выравниванием r и t .

4. Применение теоремы Байеса

На основе теоремы Байеса обновляется апостериорная вероятность $P(t | r)$, которая выражается как:

$$P(t | r) = \frac{P(r | t) \cdot F(t)}{\sum_{t \in T} P(r | t) \cdot F(t)}. \quad (1.8)$$

5. Обновление вектора композиции

После вычисления $P(t | r)$ для всех t и r , композиция таксонов обновляется следующим образом:

$$F(t) = \frac{\sum_{r \in R} P(t | r)}{|R|}, \quad (1.9)$$

где R – множество всех чтений.

6. Вычисление функции правдоподобия

Полная правдоподобность для текущей оценки F вычисляется как:

$$L(R) = \sum_{r \in R} \log \left(\sum_{t \in T} P(r | t) \cdot F(t) \right). \quad (1.10)$$

Если увеличение полной правдоподобности между итерациями превышает заданный порог (> 0.01), то процесс продолжается. В противном случае алгоритм завершается.

7. Удаление шумов

Для удаления таксонов с низкой вероятностью применяется пороговое значение. Если вероятность $F(t)$ для таксона t ниже заданного порога *threshold*, то:

$$F(t) = 0, \quad \text{если } F(t) < \text{threshold}.$$

После завершения итераций и удаления шумов производится окончательная оценка композиции, которая используется для интерпретации таксономического состава.

Эти шаги повторяются до тех пор, пока изменение правдоподобности между итерациями не станет незначительным. Окончательный результат представляет собой оценку таксономического состава.

1.7.3. Метагеномный биннинг

Метагеномный биннинг (от англ. binning – «распределение по корзинам») направлен на распределение последовательностей (прочтений или контигов), полученных из сложных микробных сообществ, по группам, отражающим их таксономическую или геномную принадлежность. Основная цель биннинга – восстановление геномов отдельных организмов из метагеномного пула, что особенно важно в контексте изучения некультивируемых микроорганизмов и оценки их функциональной роли в экосистеме.

Биннинг на основе внутренних признаков реализуется путём кластеризации последовательностей на основе их собственных характеристик, таких как GC-состав, частоты k -меров, глубина покрытия и др. Он особенно полезен при изучении ранее неописанных микробных таксонов.

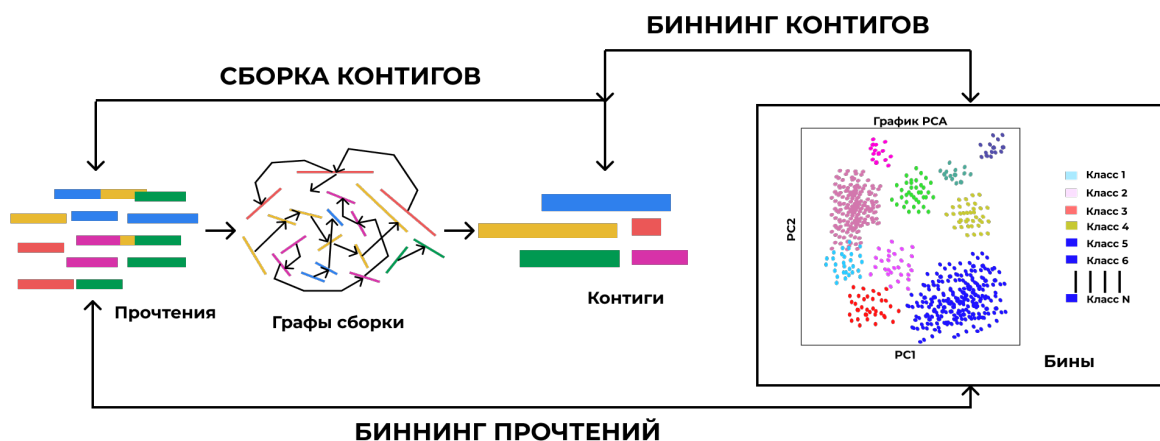


Рис. 6. Приведены два подхода к метагеномному биннингу. Первый из них предполагает непосредственную классификацию или кластеризацию прочтений (нижняя стрелка). Второй подход основан на классификации или кластеризации контигов (верхняя стрелка). В обоих случаях полученные «бины» соответствуют либо кластерам, либо заранее определённым классам.

Программы MetaBAT2 [83] и MaxBin2 [84] – одни из широко используемых инструментов для метагеномного биннинга на основе внутренних признаков. Оба алгоритма используют внутренние признаки последовательностей, такие как частоты тетра nukлеотидов и глубина покрытия, но реализуют разные подходы: MetaBAT2 применяет вероятностную модель кластеризации, тогда как MaxBin2 использует алгоритм максимизации ожидания (ЕМ) с привлечением маркерных генов для оценки полноты бинов. MetaBAT2 отличается высокой скоростью и эффективностью при работе с крупными сборками, в то время как MaxBin2 более устойчив к фрагментированным данным и лучше справляется с выявлением редких организмов.

Метагеномный биннинг может быть основан на данных химических модификаций ДНК. Программа Nanodisco [37] предназначена для идентификации различных форм модификаций ДНК (включая 4mC, 5mC и 6mA) на данных нанопорового секвенирования. В основе метода лежит сравнение сигналов (токов) между нативными ДНК-библиотеками и контрольными библиотеками, полученными с помощью полногеномной амплификации

(WGA), при которой эпигенетические модификации удаляются. Это позволяет выявить локальные отклонения в электрическом сигнале, возникающие вблизи метилированных оснований. Эти сигнальные отклонения агрегируются и нормализуются, формируя так называемые сигнатуры модификаций – профили распределения токов в окрестностях потенциальных модификаций. Далее сигнатуры используются для обучения мультиклассовых моделей машинного обучения, включая алгоритмы k-ближайших соседей, случайный лес и нейросетевые классификаторы, которые позволяют одновременно определять как тип модификации, так и её точное положение в последовательности. В рамках оригинального исследования было использовано 46 хорошо охарактеризованных мотивов, содержащих модификации, из семи бактериальных видов, на основе которых были построены обучающие выборки, охватывающие широкий спектр последовательностных контекстов.

Дополнительным преимуществом Nanodisco является его способность к *de novo* обнаружению мотивов, связанных с модификациями, в ранее неаннотированных организмах, включая сложные микробные сообщества. Для этого инструмент использует объединение p-значений в скользящих окнах, пиковый анализ и последующий поиск мотивов с использованием MEME Suite [85]. Это делает возможным не только идентификацию модификаций, но и последующую реконструкцию ландшафта модификаций без предварительных знаний о геномной структуре организма. В контексте метагеномных сборок Nanodisco демонстрирует уникальные возможности: он позволяет кластеризовать контиги на основе сходства их профилей модификаций (дополненная версия модификационного биннинга), что особенно актуально для ассоциации мобильных генетических элементов с их хозяевами и для выявления химерных или ошибочно собранных участков сборки. Инструмент эффективно компенсирует ограниченность статистической мощности при работе с фрагментированными контигами путём агрегации сигнала по множеству мотивационных контекстов и использования обученных классификаторов.

1.8. Анализ модификаций нуклеиновых кислот

1.8.1. Программный пакет *modkit motif* для поиска и анализа мотивов связанных с модификациями ДНК

Инструмент *modkit motif* [86] создан компанией ONT и предназначен для выявления поиска мотивов в геноме, которые статистически связаны с высокой степенью модификации оснований. На вход программа получает файл формата BEDMethyl. Работа алгоритма начинается с классификации геномных позиций по уровням модификации (высокий, средний или низкий), используя заданные пороги. Затем из референсной последовательности извлекаются короткие участки, окружающие каждое модифицированное основание, и на их основе формируются потенциальные мотивы. Полученные кандидаты последовательно уточняются и оцениваются на основе статистических критериев, таких как частота их встречаемости среди высоко модифицированных позиций и значимость отличий от слабо модифицированных регионов.

Алгоритм имеет гибкие настройки, которые позволяют выявлять как распространённые, так и редкие мотивы. Кроме того, предусмотрена возможность оценки и уточнения уже известных мотивов, что полезно при сравнении экспериментальных данных с ранее описанными образцами. Результаты анализа предоставляются в виде подробных таблиц, доступных как в машиночитаемом формате для дальнейшей автоматизированной обработки, так и в удобном для интерпретации пользователем графическом виде.

1.8.2. Программный пакет *MEME Suite* для поиска и анализа мотивов

Программа MEME [85] (Multiple Expectation Maximization for Motif Elicitation) используется для поиска консервативных мотивов в последовательностях нуклеиновых кислот и белков, таких как сайты связывания транскрипционных факторов или домены взаимодействия белков. Размер искомого мотива пользователь указывает в параметрах запуска программы. Алгоритм MEME основан на применении метода максимизации ожидания (Expectation-Maximization, EM) для построения и уточнения вероятностной модели, описывающей набор консервативных мотивов в исследуемом наборе последовательностей.

На первом этапе анализа MEME принимает на вход группу неродственных последовательностей, которые предположительно содержат общий мотив. Пользователь задаёт параметры, такие как минимальная и максимальная длина искомого мотива, количество предполагаемых мотивов и другие параметры. После этого MEME выполняет итеративный процесс: на шаге ожидания (Expectation, E-step) программа оценивает вероятность того, что мотив находится на каждой возможной позиции в каждой из предоставленных последовательностей, используя текущую вероятностную модель. На шаге максимизации (Maximization, M-step) MEME обновляет параметры своей модели, чтобы максимизировать вероятность обнаружения мотива в последовательностях, исходя из оценок предыдущего шага.

Полученный мотив представлен в виде позиционной весовой матрицы (PWM, Position Weight Matrix), которая описывает вероятность появления каждого нуклеотидного остатка или аминокислоты на конкретной позиции мотива. MEME предоставляет визуализацию найденных мотивов в виде логотипов последовательностей (sequence logos), которые графически показывают степень консервативности каждого символа на разных позициях.

Помимо поиска мотивов без пропусков, программа MEME Suite включает алгоритм GLAM2, позволяющий выявлять мотивы с внутренними вставками и делециями. GLAM2 использует особую модель мотивов, в которой допускаются вариации длины и пропуски в консервативных участках, что особенно важно для анализа белков, где часто присутствуют гибкие регионы. Для каждого найденного мотива GLAM2 вычисляет специальную метрику, отражающую его достоверность, а также показывает позиционный логотип с обозначением делеций и вставок.

После обнаружения мотивов MEME Suite предоставляет несколько инструментов для их дальнейшего анализа и интерпретации. Например, программа TOMTOM выполняет сравнение найденных мотивов с базами данных уже известных мотивов, предоставляя статистическую оценку сходства и возможность определить, является ли мотив новым или соответствует известным сайтам связывания. Инструменты FIMO и MAST позволяют осуществлять поиск обнаруженных мотивов в базах данных последовательностей, выявляя дополнительные возможные места связывания или функционально значимые участки. Для мотивов, содержащих пропуски, предназначен отдельный инструмент GLAM2SCAN. Дополнительно MEME Suite включает инструмент GOMO, связывающий найденные мотивы с сигнатурами генной онтологии (Gene Ontology terms [87]), что помогает исследователю понять возможную биологическую функцию идентифицированных мотивов. GOMO оценивает, как часто найденный мотив встречается в регуляторных областях генов, связанных с определёнными биологическими процессами, функциями или компонентами клетки, согласно данным генной онтологии.

1.9. Аннотация геномов

В разделе рассматриваются основные методы аннотации геномов, охватывающие как эукариотические, так и прокариотические организмы. Описаны алгоритмы и

программные инструменты, применяющиеся для предсказания генов, идентификации tandemных повторов и рибосомных РНК. Рассмотрено использование скрытых марковских моделей, динамического программирования и структурированных хеш-таблиц, а также возможность интеграции экспериментальных данных для повышения точности аннотаций.

1.9.1. *Ab initio* предсказание генов в геномах эукариот с помощью HMM и программы AUGUSTUS

AUGUSTUS [88] – это широко используемый инструмент для предсказания генов в эукариотических организмах, созданный на основе обобщённой скрытой марковской модели (Generalized Hidden Markov Model, GHMM).

GHMM повсеместно используется для предсказания генов в эукариотических геномах. Этот подход основан на вероятностной оценке последовательности ДНК, где модель представляет собой набор скрытых состояний, соответствующих различным функциональным элементам гена, таким как экзоны, интроны, межгенные области, старт- и стоп-кодоны, 5' и 3'-нетранслируемые регионы. Каждое состояние в модели характеризуется определёнными вероятностями эмиссии и перехода, отражающими типичные признаки реальных генов конкретного вида организмов. GHMM позволяет учитывать более сложную биологическую информацию, чем классические скрытые марковские модели, в том числе переменную длину состояний (например, экзонов и интронов), что значительно повышает точность предсказания структуры гена. В процессе предсказания используется алгоритм Витерби, который идентифицирует наиболее вероятную последовательность состояний для заданной геномной области.

В отличие от большинства других *ab initio* предсказателей, которые ограничиваются одной формой транскрипта на ген, AUGUSTUS предлагает уникальную возможность моделирования альтернативного сплайсинга и предсказывает сразу несколько возможных транскриптов для одного и того же гена. Это особенно важно, поскольку в реальности значительная часть эукариотических генов экспрессируется в виде нескольких альтернативных мРНК, что приводит к формированию различных белковых изоформ с различными функциями.

Алгоритмически AUGUSTUS разбивает входную последовательность на вероятные функциональные участки, используя параметры модели, обученной на конкретном биологическом виде. Программа определяет, с какой вероятностью та или иная часть последовательности принадлежит экзону, интрону или межгенному региону, после чего формирует множество возможных комбинаций этих сегментов, соответствующих различным вариантам структур генов. Один из ключевых этапов работы программы – это построение так называемого Витерби-парса, представляющего собой наиболее вероятный вариант структурной аннотации гена. Однако помимо этого, AUGUSTUS использует метод стохастического сэмплирования: программа случайным образом генерирует множество альтернативных структурных разметок (парсов), каждая из которых выбирается с вероятностью, пропорциональной её апостериорной вероятности. Затем проводится статистический анализ – подсчёт, как часто отдельные экзоны, интроны и транскрипты встречаются среди всех сгенерированных вариантов. Эти частоты интерпретируются как оценки апостериорных вероятностей, указывающие на надёжность предсказания каждого элемента.

Фильтрация альтернативных транскриптов осуществляется на основе нескольких критериев, таких как минимальная длина кодирующей последовательности, минимальная вероятность наличия экзонов и интронов, а также ограничение на общее количество транскриптов, перекрывающихся на одной и той же позиции. Таким образом, AUGUSTUS позволяет пользователю гибко настраивать чувствительность и специфичность

предсказаний: можно выбрать более строгие условия и получить точные, но ограниченные по числу предсказания, либо, наоборот, повысить чувствительность ценой увеличения числа ложноположительных вариантов.

Дополнительной возможностью, реализованной в AUGUSTUS, является поиск консервативных мотивов в предсказанных белках, с использованием регулярных выражений, включая шаблоны из базы данных PROSITE [89]. Это особенно полезно при анализе белковых семейств, где наличие определённых последовательностей аминокислотных остатков может свидетельствовать о принадлежности к функционально важной группе. Возможность автоматического скрининга по таким шаблонам позволяет исследователю быстро фильтровать интересующие варианты транскриптов из общего множества предсказаний.

Программа AUGUSTUS имеет предобученные GHMM модели для широкого круга организмов, включая человека, дрожозилу, арабидопсис, гриб *Coprinus cinereus*, паразитических червей и других. Одним из ключевых преимуществ программы является её способность интегрировать транскриптомные данные из экспериментов по секвенированию РНК для повышения точности предсказания структуры генов. Эти данные позволяют уточнить границы экзонов и интронов, подтверждая их реальную экспрессию и корректируя ошибки чисто абстрактного (*ab initio*) подхода. Сравнение с белками из близкородственных организмов помогает выявить консервативные элементы структуры генов и подтвердить функциональную состоятельность предсказанного гена. AUGUSTUS реализует это путём добавления внешних «подсказок» (hints), которые учитываются в модели GHMM и корректируют вероятности переходов между состояниями. Такой подход существенно улучшает качество предсказаний, особенно в случае малоизученных организмов, позволяя эффективнее использовать имеющиеся экспериментальные данные.

AUGUSTUS зарекомендовала себя как один из самых эффективных инструментов в арсенале современных биоинформатиков, работающих с новыми или плохо аннотированными эукариотическими геномами.

1.9.2. Поиск белковых последовательностей в геномах прокариот с помощью динамического программирования и программы Prodigal

Алгоритм Prodigal [90] (PROkaryotic DynamIc programming Genefinding ALgorithm) представляет собой полностью автоматическую систему предсказания генов у прокариот, способную находить как структуру генов, так и сайты инициации трансляции (старт-кодона). Его работа начинается с построения начального тренировочного множества генов. Для этого Prodigal анализирует весь геном, разбивая его на открытые рамки считывания (ORF) и рассчитывая GC-состав по трем позициям кодона. Для каждого ORF определяется позиция с наибольшим содержанием G/C, и на этой основе строится оценка кодирующего потенциала. Программа отбирает ORF длиной более 90 пар оснований и применяет динамическое программирование, чтобы выбрать максимально непротиворечивое покрытие генома ORF'ами, соответствующими предполагаемым генам. Далее на основе отобранного множества генов формируется более точная модель кодирующих последовательностей. Prodigal использует модель частот появления всех возможных гексамеров (6-мерных последовательностей), вычисляя логарифмическое соотношение между их частотами внутри предполагаемых генов и по всему геному. Эти значения суммируются для каждого ORF, формируя итоговый кодирующий скор. Следующим этапом является предсказание сайтов инициации трансляции. Для каждого возможного старт-кодона (ATG, GTG, TTG) программа анализирует наличие рибосомных связывающих сайтов (RBS), преимущественно основанных на последовательностях Шайна – Дальгарно. Строятся так называемые «бины» – группы

RBS-мотивов с определённым расстоянием (спейсером) до старт-кодона. Для каждого старта рассчитываются баллы, отражающие тип кодона, наличие RBS и структуру региона выше течению (upstream). Этот процесс повторяется итеративно: после оценки RBS-признаков пересматриваются позиции стартов, пересчитываются частоты и логарифмические веса, пока предсказания не стабилизируются.

После определения всех весов проводится второй проход динамического программирования с учетом более точной информации. Теперь балл каждого гена учитывает не только кодирующий потенциал, но и параметры старта. В рамках динамического программирования учитываются как одиночные ORF'ы, так и перекрывающиеся гены: допускается до 60 п.н. перекрытия на одной цепи и до 200 п.н. – на противоположных цепях. Также учитываются расстояния между генами, соответствующие типичным оперонным структурам.

По завершении второго прохода Prodigal удаляет гены с отрицательными суммарными скорингами. Короткие гены (< 250 п.н.) получают штраф, пропорциональный их длине, чтобы снизить число ложных положительных предсказаний. Программа также использует специальные эвристики, чтобы скорректировать выбор между близко расположенными стартами, полагаясь при этом только на стартовые признаки (например, если два старта расположены ближе 15 п.н. друг к другу).

В финале алгоритм возвращает координаты предсказанных генов, белковые последовательности и, при необходимости, подробную информацию о каждом рассмотренном старте. Prodigal может работать в двух режимах: либо полностью автоматически, без участия пользователя, либо в два этапа – сначала обучаясь на входной последовательности, а затем проводя предсказание. Программа отличается высокой скоростью и точностью, способна проанализировать бактериальный геном за секунды и широко применяется в автоматических аннотациях микробных геномов.

1.9.3. Поиск tandemных повторов в геномах с помощью программы RPTRF

RPTRF [91] (Rapid Perfect Tandem Repeat Finder) – это программа для поиска tandemных повторов в последовательностях ДНК. Такие повторы представляют собой участки ДНК, где один и тот же фрагмент повторяется несколько раз подряд без изменений. Обнаружение этих повторов важно для диагностики генетических заболеваний [92], изучения эволюционных процессов [93] и судебно-медицинской экспертизы [94].

На первом этапе работы программа выполняет индексирование входной последовательности. Для этого применяется структура данных под названием Sequence Hash Table (SHT), представляющая собой хеш-таблицу. С помощью скользящего окна длиной четыре символа программа последовательно проходит по всей ДНК-последовательности и на каждом шаге получает очередную 4-буквенную подстроку. Каждая такая подстрока преобразуется в бинарное значение (по два бита на символ), что позволяет использовать её в качестве индекса строки хеш-таблицы. В соответствующую строку таблицы добавляется позиция в последовательности, где встретилась данная подстрока. Таким образом, хеш-таблица собирает все повторяющиеся позиции коротких фрагментов последовательности.

После построения хеш-таблицы начинается второй этап – поиск tandemных повторов. Для этого программа просматривает каждый связанный список, прикрепленный к строкам таблицы. Внутри списка анализируются расстояния между позициями. Если обнаруживается, что несколько позиций идут с одинаковыми промежутками (например, через каждые 4 символа), это означает, что в последовательности с этой позиции начинается tandemный повтор. Количество повторов, длина мотива и общая длина

повтора рассчитываются на этом шаге.

Как только потенциальный повтор найден, программа приступает к его проверке на уникальность и корректность. Этот шаг выполняется с использованием структуры Tandem Repeat Interval Tree (TRIT). Каждый тандемный повтор рассматривается как интервал в последовательности. Если новый интервал уже полностью содержится в другом ранее найденном повторе, он отбрасывается как артефакт. Если же новый интервал полностью содержит уже существующие повторы, они удаляются из дерева, и новый интервал добавляется на их место. Если интервал не пересекается ни с одним из существующих – он просто добавляется в TRIT. Таким образом, дерево TRIT обеспечивает фильтрацию лишних или дублирующихся повторов.

Завершив поиск и фильтрацию, программа переходит к формированию окончательного результата. Из дерева TRIT с помощью симметричного обхода извлекается отсортированный список всех найденных тандемных повторов. Каждый повтор в списке содержит информацию о начальной и конечной позиции, длине мотива, количестве повторов и общем размере.

Алгоритм RPTRF отличается высокой скоростью за счёт минимизации количества сравнений символов и за счёт применения эффективных структур данных. Его производительность позволяет анализировать большие эукариотические геномы за короткое время, что делает его пригодным для использования в лабораториях с ограниченными вычислительными ресурсами.

1.9.4. Поиск генов рибосомной РНК в геномах с помощью HMM и программы *rRNAfinder*

rRNAFinder – это инструмент для поиска и классификации генов рибосомной РНК (rRNA) в геномных последовательностях. Он входит в состав пакета MetaErg [95] и использует скрытые марковские модели из программы nhmmer [96]. Работа rRNAFinder начинается с построения профилей HMM для рибосомных РНК, используя выравнивания из базы данных Rfam [97]. Для этого загружается файл rfam.seed.gz, из которого извлекаются выравнивания генов rRNA отдельно для трёх доменов жизни: бактерий, архей и эукариот. Эти выравнивания преобразуются в профили HMM с помощью программы hmmbuild из пакета HMMER [98].

После подготовки моделей инструмент переходит к основному этапу – поиску рибосомных РНК в последовательностях контингов. С этой целью nhmmer сопоставляет каждый профиль HMM с входными последовательностями, чтобы выявить участки, соответствующие генам 5S, 5.8S, 16S, 18S, 23S и 28S рРНК. В случае, если на один и тот же участок последовательности претендуют модели из разных доменов, выбирается результат с наименьшим значением E-value. Если значения E-value совпадают, сохраняются все лучшие предсказания.

На финальном этапе rRNAFinder классифицирует найденные участки, используя blastn и полные базы данных SILVA SSU и LSU [99]. Это позволяет установить точную таксономическую принадлежность каждой найденной рРНК. Результатом работы инструмента является аннотированный список рРНК-генов с указанием их типа, локализации и таксономического контекста.

ЗАКЛЮЧЕНИЕ

Нанопоровое секвенирование – это перспективный и стремительно развивающийся метод омиксных исследований, дающий уникальные возможности, такие как чтение сверхдлинных фрагментов ДНК, прямое определение эпигенетических модификаций, а также окислительных повреждений, и получение данных в реальном времени. Однако

эффективное использование этой технологии требует знания специализированных программ и алгоритмов, специфика которых отличается от аналогичных программ для коротких прочтений, что представляет сложность для начинающих исследователей.

Настоящий обзор систематизирует ключевые этапы анализа нанопоровых данных в геномных, эпигеномных и метабеномных исследованиях, а также позволяет читателю понять, какие методы и программы подходят для решения конкретных задач. В обзоре также рассмотрены современные алгоритмические решения, что особенно важно для глубокого понимания того, каким образом получаются те или иные результаты, а значит, и их достоверность.

По итогам работы предложена схема алгоритма для обработки данных нанопорового секвенирования, адаптированный для задач геномики, метагеномики и эпигеномики (рисунок 7).

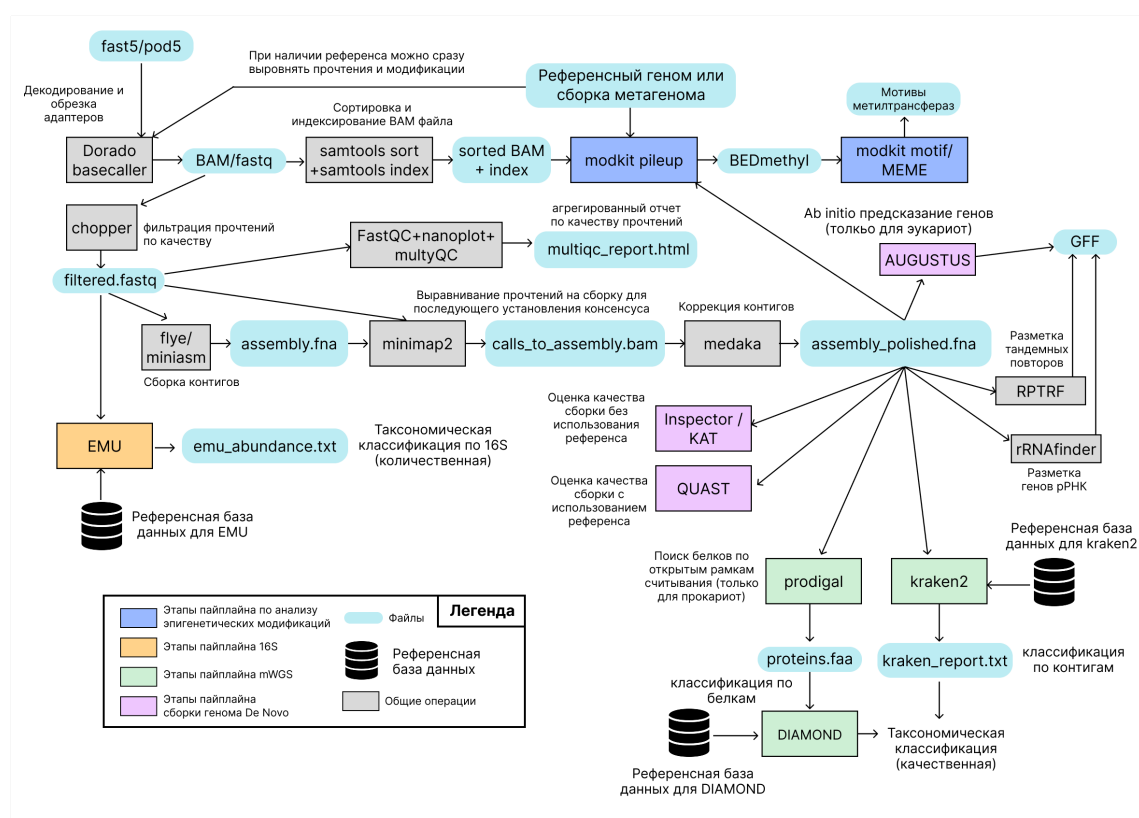


Рис. 7. Схема конвейера анализа данных нанопорового секвенирования в задачах геномики, эпигеномики и метагеномики. [Полноразмерное изображение](#).

Таким образом, этот обзор может служить отправной точкой для исследователей, начинающих работать с нанопоровым секвенированием, и способствует формированию обоснованных стратегий анализа в различных омиксных экспериментах.

Исследование, посвященное возможности определения нанопоровым секвенированием окислительной модификации гуанина (8oxG), выполнено за счет гранта Российского научного фонда № 25-14-00126, <https://rscf.ru/project/25-14-00126/>. Оставшаяся часть исследования выполнена в рамках научно-образовательной школы МГУ «Молекулярные технологии живых систем и синтетическая биология» (проект 23-Ш04-45).

СПИСОК ЛИТЕРАТУРЫ

1. Uhlen M., Quake S.R. Sequential sequencing by synthesis and the next-generation sequencing revolution. *Trends in Biotechnology*. 2023. V. 41. No. 12. P. 1565–1572. doi: [10.1016/j.tibtech.2023.06.007](https://doi.org/10.1016/j.tibtech.2023.06.007)
2. Szakállas N., Bartak B.K., Valcz G., Nagy Z.B., Takács I., Molnár B. Can long-read sequencing tackle the barriers, which the next-generation could not? A review. *Pathology and Oncology Research*. 2024. V. 30. doi: [10.3389/pore.2024.1611676](https://doi.org/10.3389/pore.2024.1611676)
3. Pervez M.T., Hasnain M.J.u., Abbas S.H., Moustafa M.F., Aslam N., Shah S.S.M. [Retracted] A Comprehensive Review of Performance of Next-Generation Sequencing Platforms. *BioMed Research International*. 2022. V. 2022. No. 1. doi: [10.1155/2022/3457806](https://doi.org/10.1155/2022/3457806)
4. Schloissnig S., Pani S., Rodriguez-Martin B., Ebler J., Hain C., Tsapalou V., Söylev A., Hüther P., Ashraf H., Prodanov T., et al. Long-read sequencing and structural variant characterization in 1, 019 samples from the 1000 Genomes Project. 2024. doi: [10.1101/2024.04.18.590093](https://doi.org/10.1101/2024.04.18.590093)
5. Lemay M., Sibbesen J.A., Torkamaneh D., Hamel J., Levesque R.C., Belzile F. Combined use of Oxford Nanopore and Illumina sequencing yields insights into soybean structural variation biology. *BMC Biology*. 2022. V. 20. No. 1. doi: [10.1186/s12915-022-01255-w](https://doi.org/10.1186/s12915-022-01255-w)
6. Dyshlovoy S.A., Paigin S., Afflerbach A., Lobermeyer A., Werner S., Schüller U., Bokemeyer C., Schuh A.H., Bergmann L., von Amsberg G., et al. Applications of Nanopore sequencing in precision cancer medicine. *International Journal of Cancer*. 2024. V. 155. No. 12. P. 2129–2140. doi: [10.1002/ijc.35100](https://doi.org/10.1002/ijc.35100)
7. Scarano C., Veneruso I., De Simone R.R., Di Bonito G., Secondino A., D'Argenio V. The Third-Generation Sequencing Challenge: Novel Insights for the Omic Sciences. *Biomolecules*. 2024. V. 14. No. 5. P. 568. doi: [10.3390/biom14050568](https://doi.org/10.3390/biom14050568)
8. Zhang T., Li H., Jiang M., Hou H., Gao Y., Li Y., Wang F., Wang J., Peng K., Liu Y. Nanopore sequencing: flourishing in its teenage years. *Journal of Genetics and Genomics*. 2024. V. 51. No. 12. P. 1361–1374. doi: [10.1016/j.jgg.2024.09.007](https://doi.org/10.1016/j.jgg.2024.09.007)
9. Ermini L., Driguez P. The Application of Long-Read Sequencing to Cancer. *Cancers*. 2024. V. 16. No. 7. P. 1275. doi: [10.3390/cancers16071275](https://doi.org/10.3390/cancers16071275)
10. Zhang J., Yang T., Xie Z., Ren Z., Shi L., Yan J., Ni M. Rapid sequencing and identification for 18-STRs long amplicon panel using portable devices and nanopore sequencer. *Genomics*. 2025. V. 117. No. 1. P. 110970. doi: [10.1016/j.ygeno.2024.110970](https://doi.org/10.1016/j.ygeno.2024.110970)
11. Park S.Y., Faraci G., Ganesh K., Dubé M.P., Lee H.Y. Portable Nanopore sequencing solution for next-generation HIV drug resistance testing. *Journal of Clinical Virology*. 2024. V. 171. P. 105639. doi: [10.1016/j.jcv.2024.105639](https://doi.org/10.1016/j.jcv.2024.105639)
12. Delahaye C., Nicolas J. Sequencing DNA with nanopores: Troubles and biases. *PLOS ONE*. 2021. V. 16. No. 10. P. e0257521. doi: [10.1371/journal.pone.0257521](https://doi.org/10.1371/journal.pone.0257521)
13. Magi A., Semeraro R., Mingrino A., Giusti B., D'Aurizio R. Nanopore sequencing data analysis: state of the art, applications and challenges. *Briefings in Bioinformatics*. 2017. doi: [10.1093/bib/bbx062](https://doi.org/10.1093/bib/bbx062)
14. Hardwick S.A., Deveson I.W., Mercer T.R. Reference standards for next-generation sequencing. *Nature Reviews Genetics*. 2017. V. 18. No. 8. P. 473–484. doi: [10.1038/nrg.2017.44](https://doi.org/10.1038/nrg.2017.44)
15. Gargis A.S., Kalman L., Bick D.P., da Silva C., Dimmock D.P., Funke B.H., Gowrisankar S., Hegde M.R., Kulkarni S., Mason C.E., et al. Good laboratory practice for clinical next-generation sequencing informatics pipelines. *Nature Biotechnology*. 2015. V. 33. No. 7. P. 689–693. doi: [10.1038/nbt.3237](https://doi.org/10.1038/nbt.3237)
16. Endrullat C., Glökler J., Franke P., Frohme M. Standardization and quality management in

- next-generation sequencing. *Applied and Translational Genomics*. 2016. V. 10. P. 2–9. doi: [10.1016/j.atg.2016.06.001](https://doi.org/10.1016/j.atg.2016.06.001)
17. Espinosa E., Bautista R., Larrosa R., Plata O. Advancements in long-read genome sequencing technologies and algorithms. *Genomics*. 2024. V. 116. No. 3. P. 110842. doi: [10.1016/j.ygeno.2024.110842](https://doi.org/10.1016/j.ygeno.2024.110842)
18. Kim C., Pongpanich M., Porntaveetus T. Unraveling metagenomics through long-read sequencing: a comprehensive review. *Journal of Translational Medicine*. 2024. V. 22. No. 1. doi: [10.1186/s12967-024-04917-1](https://doi.org/10.1186/s12967-024-04917-1)
19. Genner R., Akeson S., Meredith M., Jerez P.A., Malik L., Baker B., Miano-Burkhardt A., Paten B., Billingsley K.J., Blauwendraat C., et al. Assessing methylation detection for primary human tissue using Nanopore sequencing. *bioRxiv*. 2024. doi: [10.1101/2024.02.29.581569](https://doi.org/10.1101/2024.02.29.581569)
20. De Coster W., Weissensteiner M.H., Sedlazeck F.J. Towards population-scale long-read sequencing. *Nature Reviews Genetics*. 2021. V. 22. No. 9. P. 572–587. doi: [10.1038/s41576-021-00367-3](https://doi.org/10.1038/s41576-021-00367-3)
21. Nicholls S.M., Quick J.C., Tang S., Loman N.J. Ultra-deep, long-read nanopore sequencing of mock microbial community standards. *GigaScience*. 2019. V. 8. No. 5. doi: [10.1093/gigascience/giz043](https://doi.org/10.1093/gigascience/giz043)
22. Koranne S. Hierarchical Data Format 5 : HDF5. 2010. P. 191–200. doi: [10.1007/978-1-4419-7719-9_10](https://doi.org/10.1007/978-1-4419-7719-9_10)
23. GitHub - nanoporetech/pod5-file-format: Pod5: a high performance file format for nanopore reads. — github.com. URL: <https://github.com/nanoporetech/pod5-file-format> (accessed 24.10.2025).
24. Liu Y., Rosikiewicz W., Pan Z., Jillette N., Wang P., Taghbalout A., Foox J., Mason C., Carroll M., Cheng A., et al. DNA methylation-calling tools for Oxford Nanopore sequencing: a survey and human epigenome-wide evaluation. *Genome Biology*. 2021. V. 22. No. 1. doi: [10.1186/s13059-021-02510-z](https://doi.org/10.1186/s13059-021-02510-z)
25. Pagès-Gallego M., van Soest D.M.K., Besselink N.J.M., Straver R., Keijer J.P., Vermeulen C., Marozzi A., van Roosmalen M.J., van Boxtel R., Burgering B.M.T., et al. Direct detection of 8-oxo-dG using nanopore sequencing. 2024. doi: [10.1101/2024.05.17.594638](https://doi.org/10.1101/2024.05.17.594638)
26. Schreiber J., Karplus K. Analysis of nanopore data using hidden Markov models. *Bioinformatics*. 2015. V. 31. No. 12. P. 1897–1903. doi: [10.1093/bioinformatics/btv046](https://doi.org/10.1093/bioinformatics/btv046)
27. David M., Dursi L.J., Yao D., Boutros P.C., Simpson J.T. Nanocall: an open source basecaller for Oxford Nanopore sequencing data. *Bioinformatics*. 2016. V. 33. No. 1. P. 49–55. doi: [10.1093/bioinformatics/btw569](https://doi.org/10.1093/bioinformatics/btw569)
28. Wick R., Judd L.M., Holt K.E. Comparison Of Oxford Nanopore Basecalling Tools. 2018. doi: [10.5281/ZENODO.1188469](https://doi.org/10.5281/ZENODO.1188469)
29. Graves A. Connectionist Temporal Classification. 2012. P. 61–93. doi: [10.1007/978-3-642-24797-2_7](https://doi.org/10.1007/978-3-642-24797-2_7)
30. Wick R., Judd L.M., Holt K.E. Comparison Of Oxford Nanopore Basecalling Tools. 2018. doi: [10.5281/ZENODO.1188469](https://doi.org/10.5281/ZENODO.1188469)
31. GitHub - nanoporetech/flappie: Flip-flop basecaller for Oxford Nanopore reads — github.com. URL: <https://github.com/nanoporetech/flappie> (accessed 24.10.2025).
32. Schuster M., Paliwal K. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*. 1997. V. 45. No. 11. P. 2673–2681. doi: [10.1109/78.650093](https://doi.org/10.1109/78.650093)
33. Xu Z., Mai Y., Liu D., He W., Lin X., Xu C., Zhang L., Meng X., Mafofo J., Zaher W.A., et al. Fast-bonito: A faster deep learning based basecaller for nanopore sequencing. *Artificial Intelligence in the Life Sciences*. 2021. V. 1. P. 100011. doi: [10.1016/j.ailesci.2021.100011](https://doi.org/10.1016/j.ailesci.2021.100011)
34. Technologies O.N. Dorado: A resource-efficient basecaller with an advanced decoding

- algorithm. URL: <https://github.com/nanoporetech/dorado> (accessed 24.10.2025).
35. Ahsan M.U., Gouur A., Chan J., Zhou W., Wang K. A signal processing and deep learning framework for methylation detection using Oxford Nanopore sequencing. *Nature Communications*. 2024. V. 15. No. 1. doi: [10.1038/s41467-024-45778-y](https://doi.org/10.1038/s41467-024-45778-y)
 36. Ni P., Huang N., Zhang Z., Wang D., Liang F., Miao Y., Xiao C., Luo F., Wang J. DeepSignal: detecting DNA methylation state from Nanopore sequencing reads using deep-learning. *Bioinformatics*. 2019. V. 35. No. 22. P. 4586–4595. doi: [10.1093/bioinformatics/btz276](https://doi.org/10.1093/bioinformatics/btz276)
 37. Tourancheau A., Mead E.A., Zhang X., Fang G. Discovering multiple types of DNA methylation from bacteria and microbiome using nanopore sequencing. *Nature Methods*. 2021. V. 18. No. 5. P. 491–498. doi: [10.1038/s41592-021-01109-3](https://doi.org/10.1038/s41592-021-01109-3)
 38. Sergeev A.V., Malyshev D.P., Genatullina A.I., Pavlova G.V., Gromova E.S., Zvereva M.I. Single-Molecule Nanopore Sequencing of the CpG Island from the Promoter of O6-Methylguanine-DNA Methyltransferase Provides Insights into the Mechanism of De Novo Methylation of G/C-Rich Regions. *Epigenomes*. 2025. V. 9. No. 1. P. 4. doi: [10.3390/epigenomes9010004](https://doi.org/10.3390/epigenomes9010004)
 39. Pagès-Gallego M., van Soest D.M.K., Besselink N.J.M., Straver R., Keijer J.P., Vermeulen C., Marcozzi A., van Roosmalen M.J., van Boxtel R., Burgering B.M.T., et al. Direct detection of 8-oxo-dG using nanopore sequencing. *Nature Communications*. 2025. V. 16. No. 1. doi: [10.1038/s41467-025-60391-3](https://doi.org/10.1038/s41467-025-60391-3)
 40. GitHub - nanoporetech/remora: Methylation/modified base calling separated from basecalling. — github.com. URL: <https://github.com/nanoporetech/remora> (accessed 24.10.2025).
 41. Ewing B., Hillier L., Wendl M.C., Green P. Base-Calling of Automated Sequencer Traces Using Phred. I. Accuracy Assessment. *Genome Research*. 1998. V. 8. No. 3. P. 175–185. doi: [10.1101/gr.8.3.175](https://doi.org/10.1101/gr.8.3.175)
 42. White R., Pellefigues C., Ronchese F., Lamiabie O., Eccles D. Investigation of chimeric reads using the MinION. *F1000Research*. 2017. V. 6. P. 631. doi: [10.12688/f1000research.11547.2](https://doi.org/10.12688/f1000research.11547.2)
 43. De Coster W., Rademakers R. NanoPack2: population-scale evaluation of long-read sequencing data. *Bioinformatics*. 2023. V. 39. No. 5. doi: [10.1093/bioinformatics/btad311](https://doi.org/10.1093/bioinformatics/btad311)
 44. Chen S., Zhou Y., Chen Y., Gu J. fastp: an ultra-fast all-in-one FASTQ preprocessor. *Bioinformatics*. 2018. V. 34. No. 17. P. i884–i890.
 45. Babraham Bioinformatics - FastQC A Quality Control tool for High Throughput Sequence Data. URL: <https://www.bioinformatics.babraham.ac.uk/projects/fastqc/> (accessed 24.10.2025).
 46. Babraham Bioinformatics - FastQ Screen. URL: <https://www.bioinformatics.babraham.ac.uk/projects/fastq-screen/> (accessed 24.10.2025).
 47. Ewels P., Magnusson M., Lundin S., Käller M. MultiQC: summarize analysis results for multiple tools and samples in a single report. *Bioinformatics*. 2016. V. 32. No. 19. P. 3047–3048.
 48. Li Z., Chen Y., Mu D., Yuan J., Shi Y., Zhang H., Gan J., Li N., Hu X., Liu B., et al. Comparison of the two major classes of assembly algorithms: overlap-layout-consensus and de-bruijn-graph. *Briefings in Functional Genomics*. 2011. V. 11. No. 1. P. 25–37. doi: [10.1093/bfpg/blr035](https://doi.org/10.1093/bfpg/blr035)
 49. Huson D.H., Reinert K., Myers E.W. The greedy path-merging algorithm for contig scaffolding. *Journal of the ACM*. 2002. V. 49. No. 5. P. 603–615. doi: [10.1145/585265.585267](https://doi.org/10.1145/585265.585267)
 50. Sohn J., Nam J. The present and future of de novo whole-genome assembly. *Briefings in Bioinformatics*. 2016. P. bbw096. doi: [10.1093/bib/bbw096](https://doi.org/10.1093/bib/bbw096)

51. Miller J.R., Koren S., Sutton G. Assembly algorithms for next-generation sequencing data. *Genomics*. 2010. V. 95. No. 6. P. 315–327. doi: [10.1016/j.ygeno.2010.03.001](https://doi.org/10.1016/j.ygeno.2010.03.001)
52. Cherukuri Y., Janga S.C. Benchmarking of de novo assembly algorithms for Nanopore data reveals optimal performance of OLC approaches. *BMC Genomics*. 2016. V. 17. No. S7. doi: [10.1186/s12864-016-2895-8](https://doi.org/10.1186/s12864-016-2895-8)
53. Cosma B., Shirali Hossein Zade R., Jordan E.N., van Lent P., Peng C., Pillay S., Abeel T. Evaluating long-read de novo assembly tools for eukaryotic genomes: insights and considerations. *Gigascience*. 2022. V. 12.
54. Kolmogorov M., Yuan J., Lin Y., Pevzner P.A. Assembly of long, error-prone reads using repeat graphs. *Nat. Biotechnol.* 2019. V. 37. No. 5. P. 540–546.
55. Li H. Minimap and miniasm: fast mapping and de novo assembly for noisy long sequences. *Bioinformatics*. 2016. V. 32. No. 14. P. 2103–2110.
56. Koren S., Walenz B.P., Berlin K., Miller J.R., Bergman N.H., Phillippy A.M. Canu: scalable and accurate long-read assembly via adaptivek-mer weighting and repeat separation. *Genome Research*. 2017. V. 27. No. 5. P. 722–736. doi: [10.1101/gr.215087.116](https://doi.org/10.1101/gr.215087.116)
57. Li H. Minimap2: pairwise alignment for nucleotide sequences. *Bioinformatics*. 2018. V. 34. No. 18. P. 3094–3100.
58. Sedlazeck F.J., Rescheneder P., Smolka M., Fang H., Nattestad M., von Haeseler A., Schatz M.C. Accurate detection of complex structural variations using single-molecule sequencing. *Nature Methods*. 2018. V. 15. No. 6. P. 461–468. doi: [10.1038/s41592-018-0001-7](https://doi.org/10.1038/s41592-018-0001-7)
59. Sović I., Šikić M., Wilm A., Fenlon S.N., Chen S., Nagarajan N. Fast and sensitive mapping of nanopore sequencing reads with GraphMap. *Nature Communications*. 2016. V. 7. No. 1. doi: [10.1038/ncomms11307](https://doi.org/10.1038/ncomms11307)
60. Suzuki H., Kasahara M. Introducing difference recurrence relations for faster semi-global alignment of long sequences. *BMC Bioinformatics*. 2018. V. 19. No. S1. doi: [10.1186/s12859-018-2014-8](https://doi.org/10.1186/s12859-018-2014-8)
61. Li H., Durbin R. Fast and accurate long-read alignment with Burrows-Wheeler transform. *Bioinformatics*. 2010. V. 26. No. 5. P. 589–595. doi: [10.1093/bioinformatics/btp698](https://doi.org/10.1093/bioinformatics/btp698)
62. Camacho C., Coulouris G., Avagyan V., Ma N., Papadopoulos J., Bealer K., Madden T.L. BLAST+: architecture and applications. *BMC Bioinformatics*. 2009. V. 10. No. 1. doi: [10.1186/1471-2105-10-421](https://doi.org/10.1186/1471-2105-10-421)
63. Walker B.J., Abeel T., Shea T., Priest M., Abouelliel A., Sakthikumar S., Cuomo C.A., Zeng Q., Wortman J., Young S.K., et al. Pilon: an integrated tool for comprehensive microbial variant detection and genome assembly improvement. *PLoS One*. 2014. V. 9. No. 11. P. e112963.
64. Lee J.Y., Kong M., Oh J., Lim J., Chung S.H., Kim J., Kim J., Kim K., Yoo J., Kwak W. Comparative evaluation of Nanopore polishing tools for microbial genome assembly and polishing strategies for downstream analysis. *Sci. Rep.* 2021. V. 11. No. 1. P. 1–11.
65. GitHub - nanoporetech/medaka: Sequence correction provided by ONT Research. URL: <https://github.com/nanoporetech/medaka> (accessed 24.10.2025).
66. Rhie A., Walenz B.P., Koren S., Phillippy A.M. Merqury: reference-free quality, completeness, and phasing assessment for genome assemblies. *Genome Biology*. 2020. V. 21. No. 1. doi: [10.1186/s13059-020-02134-9](https://doi.org/10.1186/s13059-020-02134-9)
67. Chen Y., Zhang Y., Wang A.Y., Gao M., Chong Z. Accurate long-read de novo assembly evaluation with Inspector. *Genome Biology*. 2021. V. 22. No. 1. doi: [10.1186/s13059-021-02527-4](https://doi.org/10.1186/s13059-021-02527-4)
68. Mapleson D., Garcia Accinelli G., Kettleborough G., Wright J., Clavijo B.J. KAT: a K-mer analysis toolkit to quality control NGS datasets and genome assemblies. *Bioinformatics*. 2016. V. 33. No. 4. P. 574–576. doi: [10.1093/bioinformatics/btw663](https://doi.org/10.1093/bioinformatics/btw663)

69. Tran Q., Phan V. Assembling Reads Improves Taxonomic Classification of Species. *Genes*. 2020. V. 11. No. 8. P. 946.
70. Wood D.E., Salzberg S.L. Kraken: ultrafast metagenomic sequence classification using exact alignments. *Genome Biol.* 2014. V. 15. No. 3. P. 1–12.
71. Wood D.E., Lu J., Langmead B. Improved metagenomic analysis with Kraken 2. *Genome Biol.* 2019. V. 20. No. 1. P. 1–13.
72. Kim D., Song L., Breitwieser F.P., Salzberg S.L. Centrifuge: rapid and sensitive classification of metagenomic sequences. *Genome Res.* 2016. V. 26. No. 12. P. 1721–1729.
73. Menzel P., Ng K.L., Krogh A. Fast and sensitive taxonomic classification for metagenomics with Kaiju. *Nat. Commun.* 2016. V. 7. P. 11257.
74. Johnson M., Zaretskaya I., Raytselis Y., Merezhuik Y., McGinnis S., Madden T.L. NCBI BLAST: a better web interface. *Nucleic Acids Res.* 2008. V. 36. No. suppl_2. P. W5–W9.
75. Blanco-Míguez A., Beghini F., Cumbo F., McIver L.J., Thompson K.N., Zolfo M., Manghi P., Dubois L., Huang K.D., Thomas A.M., et al. Extending and improving metagenomic taxonomic profiling with uncharacterized species using MetaPhlAn 4. *Nat. Biotechnol.* 2023. V. 41. No. 11. P. 1633–1644.
76. Buchfink B., Reuter K., Drost H. Sensitive protein alignments at tree-of-life scale using DIAMOND. *Nat. Methods.* 2021. V. 18. No. 4. P. 366–368.
77. Huson D.H., Auch A.F., Qi J., Schuster S.C. MEGAN analysis of metagenomic data. *Genome Res.* 2007. V. 17. No. 3. P. 377–386.
78. Pruitt K.D., Tatusova T., Brown G.R., Maglott D.R. NCBI Reference Sequences (RefSeq): current status, new features and genome annotation policy. *Nucleic Acids Research.* 2011. V. 40. No. D1. P. D130–D135. doi: [10.1093/nar/gkr1079](https://doi.org/10.1093/nar/gkr1079)
79. Marçais G., Kingsford C. A fast, lock-free approach for efficient parallel counting of occurrences of k-mers. *Bioinformatics.* 2011. V. 27. No. 6. P. 764–770. doi: [10.1093/bioinformatics/btr011](https://doi.org/10.1093/bioinformatics/btr011)
80. Hantze H., Horton P. Effects of spaced k-mers on alignment-free genotyping. *Bioinformatics.* 2023. V. 39. No. Supplement_1. P. i213–i221. doi: [10.1093/bioinformatics/btad202](https://doi.org/10.1093/bioinformatics/btad202)
81. Karlsson F.H., Fåk F., Nookaew I., Tremaroli V., Fagerberg B., Petranovic D., Bäckhed F., Nielsen J. Symptomatic atherosclerosis is associated with an altered gut metagenome. *Nature Communications.* 2012. V. 3. No. 1. doi: [10.1038/ncomms2266](https://doi.org/10.1038/ncomms2266)
82. Curry K.D., Wang Q., Nute M.G., Tyshaieva A., Reeves E., Soriano S., Wu Q., Graeber E., Finzer P., Mendling W., et al. Emu: species-level microbial community profiling of full-length 16S rRNA Oxford Nanopore sequencing data. *Nature Methods.* 2022. V. 19. No. 7. P. 845–853. doi: [10.1038/s41592-022-01520-4](https://doi.org/10.1038/s41592-022-01520-4)
83. Kang D.D., Li F., Kirton E., Thomas A., Egan R., An H., Wang Z. MetaBAT 2: an adaptive binning algorithm for robust and efficient genome reconstruction from metagenome assemblies. *PeerJ.* 2019. V. 7. P. e7359. doi: [10.7717/peerj.7359](https://doi.org/10.7717/peerj.7359)
84. Wu Y., Simmons B.A., Singer S.W. MaxBin 2.0: an automated binning algorithm to recover genomes from multiple metagenomic datasets. *Bioinformatics.* 2015. V. 32. No. 4. P. 605–607. doi: [10.1093/bioinformatics/btv638](https://doi.org/10.1093/bioinformatics/btv638)
85. Bailey T.L., Boden M., Buske F.A., Frith M., Grant C.E., Clementi L., Ren J., Li W.W., Noble W.S. MEME SUITE: tools for motif discovery and searching. *Nucleic Acids Research.* 2009. V. 37. No. Web Server. P. W202–W208. doi: [10.1093/nar/gkp335](https://doi.org/10.1093/nar/gkp335)
86. GitHub - nanoporetech/modkit: A bioinformatics tool for working with modified bases — github.com. URL: <https://github.com/nanoporetech/modkit> (accessed 24.10.2025).
87. Aleksander S.A., Balhoff J., Carbon S., Cherry J.M., Drabkin H.J., Ebert D., Feuermann M., Gaudet P., Harris N.L., Hill D.P., et al. The Gene Ontology knowledgebase in 2023.

- GENETICS*. 2023. V. 224. No. 1. doi: [10.1093/genetics/iyad031](https://doi.org/10.1093/genetics/iyad031)
88. Stanke M., Keller O., Gunduz I., Hayes A., Waack S., Morgenstern B. AUGUSTUS: ab initio prediction of alternative transcripts. *Nucleic Acids Research*. 2006. V. 34. No. Web Server. P. W435–W439. doi: [10.1093/nar/gkl200](https://doi.org/10.1093/nar/gkl200)
 89. Sigrist C.J.A., de Castro E., Cerutti L., Cuche B.A., Hulo N., Bridge A., Bougueleret L., Xenarios I. New and continuing developments at PROSITE. *Nucleic Acids Research*. 2012. V. 41. No. D1. P. D344–D347. doi: [10.1093/nar/gks1067](https://doi.org/10.1093/nar/gks1067)
 90. Hyatt D., Chen G., LoCascio P.F., Land M.L., Larimer F.W., Hauser L.J. Prodigal: prokaryotic gene recognition and translation initiation site identification. *BMC Bioinformatics*. 2010. V. 11. No. 1. doi: [10.1186/1471-2105-11-119](https://doi.org/10.1186/1471-2105-11-119)
 91. Behboudi R., Nouri-Baygi M., Naghibzadeh M. RPTRF: A rapid perfect tandem repeat finder tool for DNA sequences. *Biosystems*. 2023. V. 226. P. 104869. doi: [10.1016/j.biosystems.2023.104869](https://doi.org/10.1016/j.biosystems.2023.104869)
 92. Ryan C.P. Tandem repeat disorders. *Evol. Med. Public Health*. 2019. V. 2019. No. 1. P. 17.
 93. Fan H., Chu J. A brief review of short tandem repeat mutation. *Genomics Proteomics Bioinformatics*. 2007. V. 5. No. 1. P. 7–14.
 94. Wyner N., Barash M., McNevin D. Forensic autosomal short tandem repeats and their potential association with phenotype. *Front. Genet.* 2020. V. 11. P. 884.
 95. Dong X., Strous M. An integrated pipeline for annotation and visualization of metagenomic contigs. *Front. Genet.* 2019. V. 10. P. 999.
 96. Wheeler T.J., Eddy S.R. nhmmer: DNA homology search with profile HMMs. *Bioinformatics*. 2013. V. 29. No. 19. P. 2487–2489.
 97. Ontiveros-Palacios N., Cooke E., Nawrocki E.P., Triebel S., Marz M., Rivas E., Griffiths-Jones S., Petrov A.I., Bateman A., Sweeney B. Rfam 15: RNA families database in 2025. *Nucleic Acids Research*. 2024. V. 53. No. D1. P. D258–D267. doi: [10.1093/nar/gkae1023](https://doi.org/10.1093/nar/gkae1023)
 98. Finn R.D., Clements J., Eddy S.R. HMMER web server: interactive sequence similarity searching. *Nucleic Acids Research*. 2011. V. 39. No. suppl. P. W29–W37. doi: [10.1093/nar/gkr367](https://doi.org/10.1093/nar/gkr367)
 99. Glöckner F.O. The SILVA Database Project: An ELIXIR core data resource for high-quality ribosomal RNA sequences. *Biodiversity Information Science and Standards*. 2019. V. 3. doi: [10.3897/biss.3.36125](https://doi.org/10.3897/biss.3.36125)

Рукопись поступила в редакцию 13.08.2025.

Переработанный вариант поступил 26.09.2025.

Дата опубликования 02.12.2025.

Tools and Algorithms for Nanopore Sequencing Data Analysis in Genomics, Metagenomics, and Epigenomics

Eremin A.A. , Sergeev A.V., Zvereva M.E., Khrenova M.G.

Lomonosov Moscow State University, Moscow, Russia

Abstract. Nanopore sequencing is a promising technology for reading long fragments of DNA and RNA. This paper reviews modern algorithms and software tools used for the analysis of nanopore sequencing data in genomics, metagenomics, and epigenomics. The article describes the algorithms behind the most widely used software tools, highlighting their strengths and limitations. The presented review is useful for researchers working with nanopore sequencing data, as well as for developers of new methods for processing bioinformatics data.

Key words: *nanopore sequencing, de novo genome assembly, genomics, epigenomics, metagenomics, long reads*