Лицей «Физико-техническая школа» им. Ж. И. Алфёрова
Санкт-Петербургского Академического университета

Курсовая работа (отчет по практике)

Реализация алгоритма поиска крупных геномных перестроек на базе сравнения близкородственных геномов

Работу выполнили:
ученик 11б класса *Ткачев Денис*
ученик 11а класса *Панюхин Никита*

Научные руководители:
Яковлев Павел Андреевич
Шугаева Татьяна Евгеньевна
Надолинский Александр Анатольевич

Место прохождения практики:
Департамент вычислительной биологии BIOCAD

Санкт-Петербург
2020

Содержание

ВВЕДЕНИЕ

В каждой клетке эукариотического организма присутствует ядро – органоид, в котором содержится информация, необходимая для построения и восстановления клетки и её составляющих. В само́м ядре содержится *геном* – совокупность наследственного материала организма.

Геном может быть зашифрован в ДНК и РНК. В большинстве случаев, в том числе и у человека, **д**езоксирибо**н**уклеиновая **к**ислота отвечает за хранение и передачу генетической информации. ДНК и РНК – это макромолекулы, состоящие из *нуклеотидов*. Каждый нуклеотид состоит из азотистого основания, сахарного остатка и фосфатной группы, однако, рассматриваемые нами, отличаются лишь азотистыми основаниями. Всего существует 5 азотистых оснований: *аденин* (**A**), *цитозин* (**C**), *тимин* (**T**), *гуанин* (**G**) и *урацил* (**U**), но в ДНК встречаются только первые четыре, а в РНК – все, кроме *тимина* (**T**). Таким образом можно представить молекулу ДНК (аналогично для РНК) в виде последовательности букв **A**, **C**, **T**, и **G** - сокращенных названий последовательно идущих нуклеотидов в одной из молекул ДНК**.** Расшифровка последовательности ДНК и РНК из биологических образцов называется *секвенированием*.

Одним из главных прорывов последних десятилетий в биологии является разработка методов секвенирования второго поколения, по-английски они объединены названием Next Generation Sequencing (NGS). Если раньше на секвенирование генома человека уходили годы, то с приходом таких платформ, как Illumina, на расшифровку стали уходить недели, а то и дни. Эти технологии имеют огромное значение как для фундаментальной биологии, так и на практике. Они стали толчком для так называемой “персонализированной медицины”: теперь мы можем расшифровать собственный геном и узнать риски развития определенных заболеваний у нас и наших детей.

Но ведь не только геном человека поддается считыванию: не менее важным открытием разработка методов NGS стала для изучения наших маленьких соседей по планете - бактерий. Они и помогают нам, проживая в нашем кишечнике, участвуя в различных процессах пищевой промышленности и много где ещё. (Масса бактерий составляет $\sim 1\ 3\%$ массы нашего тела, то есть в среднем около $1\ 2$ кг.)

Например, раньше классификация бактерий базировалась на небольших последовательностях (16S rRNA длиной примерно $1{,}5$ тысячи пар оснований). Можно было сравнить эти последовательности и кластеризовать - чем больше похожи последовательности у пары бактерий, тем большее родство они имеют. Теперь же в мировой науке сформировался тренд на построение классификации по целым геномам. Это более вычислительно затратно, зато позволяет гораздо лучше разделять близкие виды.

Быстрая расшифровка генома является также ценным оружием в борьбе с постоянно мутирующими патогенными бактериями. Чем дальше, тем больше появляется штаммов с широкой устойчивостью к антибиотикам, которые, соответственно, плохо поддаются лечению. Расшифровка и сравнение их геномов зачастую позволяет найти участки с отличиями и понять механизм приобретенной устойчивости гораздо быстрее, чем долгие и опасные биологические эксперименты на живых бактериях. Например, в работе *“Genome evolution driven by host adaptations results in a more virulent and antimicrobial-resistant Streptococcus pneumoniae serotype 14” (2009)* было произведено сравнение различных штаммов пневмонийного стрептококка и

выявлены генетические изменения, объясняющие повышенную лекарственную устойчивость новых штаммов.

Но задача сравнения геномов не так проста, как сравнение небольших последовательностей: в них могут происходить не только маленькие замены нескольких букв, но и огромные перестроения в тысячи нуклеотидов. Выявлением подобных перестроений и занимается наш алгоритм.

ПОСТАНОВКА ЗАДАЧИ

Автоматизировать сравнение геномов прокариотических бактерий для попарного выявления произошедших эволюционных мутационных изменений и построение некоторой структуры родственности для количественного ответа на вопрос о степени родства двух взятых представителей бактерий.

То есть, сама работа делится на 3 функциональные части:

- Выравнивание пар геномов
- Выявление “значимых” мутационных эволюционных изменений
- Составление структур родственности на базе сделанного попарного сравнивания (в разработке)

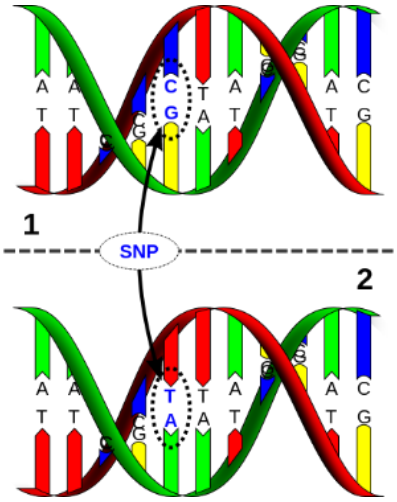
В перспективе написанный софт будет помогать в нахождении ближайших “родственников” представителя данного вида супербактерий – бактерий, для лечения которых ещё не найден соответствующий антибиотик, для попытки видоизменения антибиотиков, используемых против родственников данной бактерии для попытки лечения этой супербактерии модифицированными антибиотиками.

МЕТОДИКА

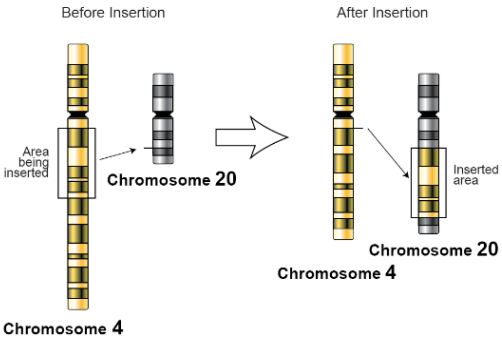
Мутации

Для начала рассмотрим мутации, которые происходят с геномами в процессе эволюции и которые встречаются нам при сравнении, чтобы иметь представление о том, какие задачи было необходимо решить в рамках этой работы:

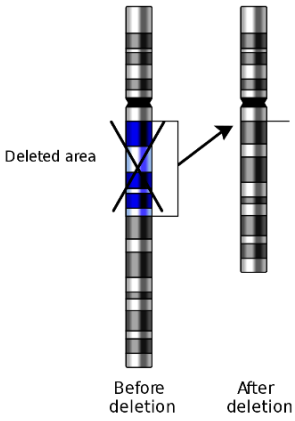
- **Мононуклеотидные замены** – отличия последовательности ДНК размером в один нуклеотид (**A**, **C**, **T** или **G**) в геноме представителей одного вида или между гомологичными участками гомологичных хромосом.



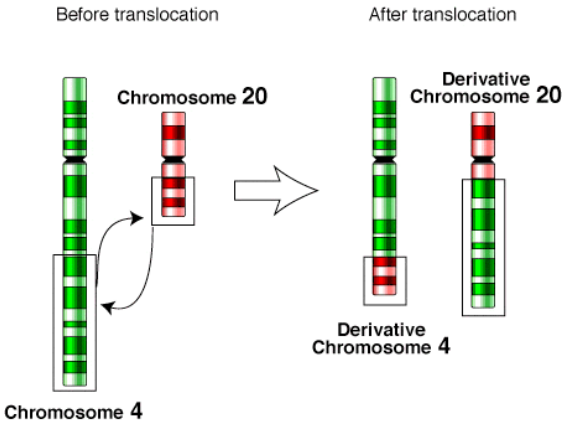
- **Инсерция (вставка)** – генетическая мутация, при которой происходит вставка одной последовательности ДНК в другую.



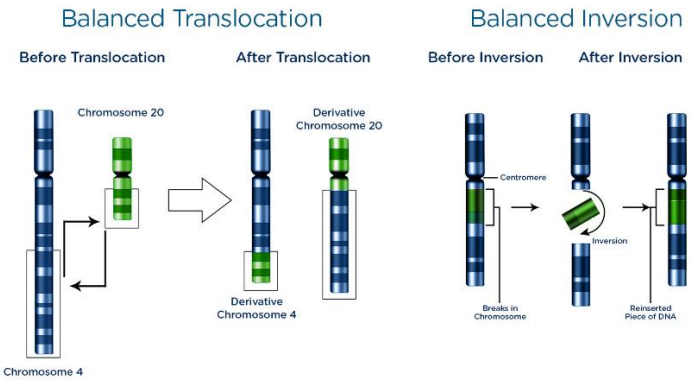
- **Делеция** – хромосомная перестройка, при которой происходит потеря участка хромосомы.



- **Транслокация** – тип хромосомных мутаций, при которых происходит перенос участка хромосомы на негомологичную хромосому.



- **Инверсия (поворот)** – хромосомная перестройка, при которой происходит поворот участка хромосомы на 180°.



Существующие инструменты

Для решения задачи выравнивания геномов существуют инструменты, такие как:

- **BLAST** – довольно неплохо справляется с выравниванием. Учитывает инверсии. То, что нам нужно для первой части работы.
- **Fasta** – аналог BLAST'а. Не учитывает инверсии.

Данные инструменты вполне выполняют поставленные задачи, но разработка собственного решения являлась частью нашего исследования, поэтому мы создали софт, производящий выравнивание двух геномов с записью результата в удобном формате для дальнейшей обработки на следующем этапе – анализе выравнивания.

Алгоритм выравнивания

Прежде чем мы начнем, введем некоторые обозначения:

При попарном выравнивании геномов у нас есть два генома, т.е. две больших последовательности нуклеотидов – символов **ACTG**. Один из геномов будем называть *query-последовательностью* (от англ. "Query" – запрос), а второй – *target-последовательностью* (от англ. "Target" – цель). Данные обозначения равносильно используются для того, чтобы отличать один геном от другого при описании алгоритмов. Теоретическую сложность алгоритмов будем обозначать по правилам асимптотического сравнения, как $O(\dots)$.

Нужно сказать, что мы работаем с большими данными, т.е. реализация наивных алгоритмов выравнивания за $O(n^3)$ и даже $O(n^2)$ будет бессмысленна, поскольку вычислительные мощности ЭВМ, которые были доступны нам во время разработки – на уровне обычных персональных компьютеров.

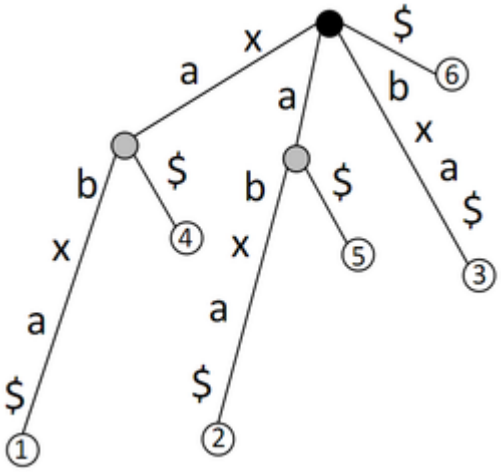
В части программирования, первое решение, которое нужно принять – выбор оптимального языка программирования. Мы выбрали язык C++ для решения задачи выравнивания. Он быстрый и удобный в плане определения математических выражений.

Второе – нужно было задуматься о том, как уменьшить сложность алгоритма. Пришлось делать некоторые вычисления приближенными, чтобы на несколько порядков уменьшить объем обрабатываемых данных, а также реализовать построение и использование структур данных, перед этим изучив некоторое количество материала по статистике, теории случайных блуждание, оптимизированному построению суффиксных деревьев и т.п.

Наиболее эффективно используемая у нас структура данных – суффиксное дерево, а, точнее, суффиксный лес, т.е. упорядоченное множество суффиксных деревьев. Давайте сначала рассмотрим структуру, на базе которой создана модель суффиксного дерева – бор:

Бор (префиксное дерево) — структура данных для хранения набора строк, представляющая из себя подвешенное дерево с символами на рёбрах. Строки получаются последовательной записью всех символов, хранящихся на рёбрах между корнем бора и вершиной конечного назначения.

Суффиксное дерево — бор, содержащий все суффиксы некоторой строки (и только их).



Черная вершина на рисунке – корень суффиксного дерева. Движение происходит строго вниз по одной из ветвей текущей вершины. Вершина, не имеющая ветвей, направленных вниз, называется листом. Суффиксное дерево устроено так, что при любом пути из вершины в любое дерево при конкатенации строк, которые записаны на ветвях, по которым происходит последовательное перемещение, получается в итоге строка, которая является суффиксом главное строки - на базе которой строится это суффиксное дерево.

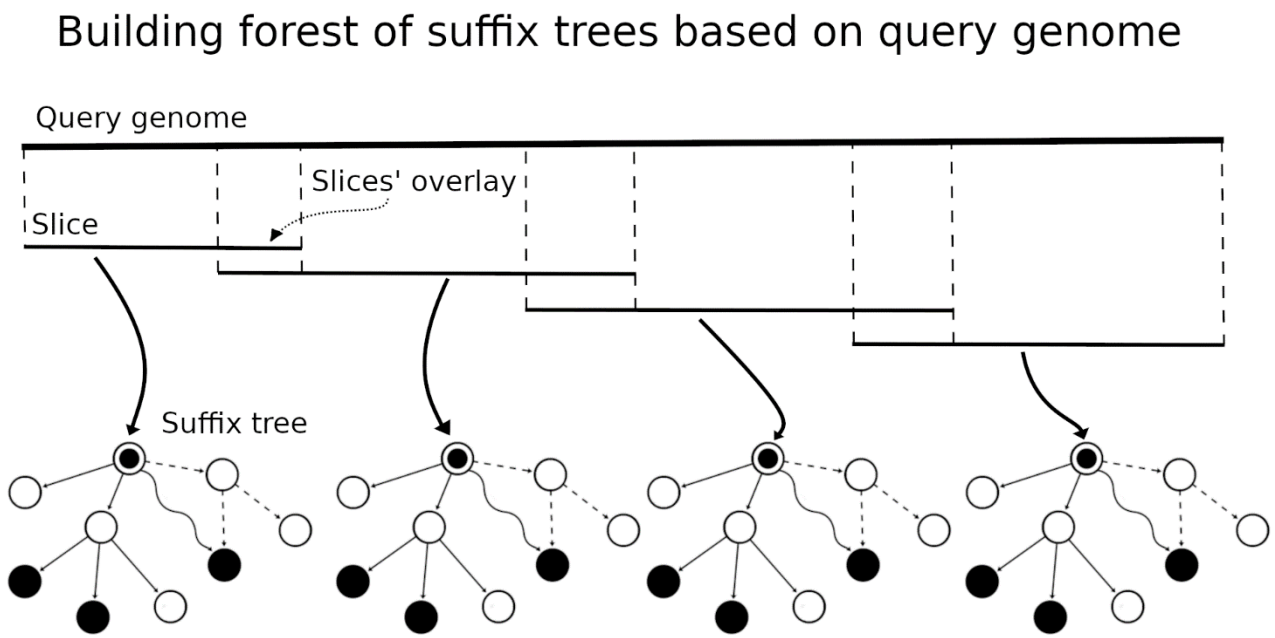
Таким образом, данная структура данных позволяет выяснять, входит ли рассматриваемая строка в строку, на базе которой строится суффиксное дерево, за $O(n)$, где n - длина строки, чье вхождение или его отсутствие мы хотим определить. Для ответа на этот вопрос достаточно спускаться из корня дерева к следующих вершинам, пока рассматриваемая строка будет “накладываться” на подстроку какой-нибудь из ветвей текущей вершины, идущих вниз.

Следует заметить, что перед использованием такой удобной структуры данных, её для начала нужно построить.

Реализация алгоритма построения сжатого суффиксного дерева без оптимизации имеет сложность $O(n^2)$. Для этого существует метод использования префиксных ссылок во время построения, что ускоряет алгоритм до $O(n \cdot log\ n)$. Но, как выяснилось во время тестирования данного алгоритма, количество префиксных ссылок получается слишком велико, что начинает занимать некоторое количество памяти, видимо, из-за того, что вершин строится очень много из-за того, что в алфавите нашей строки присутствует ровно 4 различных символа, вследствие чего алгоритм построения работает намного дольше, чем должен, сравнивая с теоретической сложностью $O(n \cdot log\ n)$.

Объяснение алгоритма построения сжатого суффиксного дерева с использованием префиксных ссылок выходит за рамки темы данного разговора, поэтому мы оставим ссылки на литературу, которой мы пользовались для реализации этого алгоритма.

В итоге получилось, что данные слишком большие, а алфавит слишком мал, для построения суффиксных деревьев за $O(n \cdot log\, n)$, а построение структуры на базе одного из геномов за $O(n^2)$ занимало слишком большое время - около месяца. Поэтому мы заменили построение одного большого суффиксного дерева на построение суффиксного дерева, т.е. мы разбили query-геном на множество подстрок, имеющих на концах общие накладывающиеся участки, чтобы на базе каждой из подстрок построить суффиксное дерево, а потом объединить все деревья в лес. Это позволило снизить сложность до $O\left(\frac{n^2}{k}\right)$, где n - длина query-генома, k - количество подстрок, на которые была разбита query-последовательность. Следует сказать, что накрадывающиеся участки последовательно строящихся деревьев имеют длину бóльшую, чем длина подстрок, чьи вхождения мы рассматривали.



Получившаяся сложность определения вхождения подстроки в суффиксный лес, т.е. вхождения подстроки из target-генома в query-геном, получилась $O(n \cdot k)$, где n - длина подстроки, а k - количество деревьев в суффиксном лесу, что достаточно быстро.

Следующим шагом target-геном разбивался на последовательно идущие, не имеющие пересечений, подстроки. Следует помнить, что работаем мы не просто с подстроками, а с геномами, поэтому нужно было учитывать ещё и возможность вхождения комплиментарной подстроки, а также её версии с инверсией. Для каждой такой подстроки алгоритм искал вхождение в query-последовательность (построенный суффиксный лес): самой подстроки, ей комплиментарной, подстроки с инверсией и комплиментарной подстроки с инверсией. Индекс каждого вхождения, а также информация о наличии инверсии или её отсутствии, записывались в файл буферного обмена, чтобы исключить лишний запуск алгоритма на выравнивание уже обработанной пары геномов.

Для лучшего понимая итога, который выдает по окончанию программа, следует показать формат буферного файла:

```
> Название и информация о первом (query) геноме

> Название и информация о втором (target) геноме
```

```
Первая_подстрока : {список индексов её вхождений в первый геном через
запятую};
{список индексов её вхождений во второй геном через запятую}

Вторая_подстрока : …
```

Пример части буферного файла:

```
> Q : >CP000766.3 Rickettsia rickettsii str. Iowa, complete genome
> T : >CP003305.1 Rickettsia rickettsii str. Brazil, complete genome
AAAAAAATAAAATATTTCTT : 232660 ; 947724
AAAAAACTTGTTTTTTTAGG : 158640 ; 692504
AAAAAAGTCTTACAATTTGC : 673520 ; 495547
AAAAAATCTTATAATTTGCA : 174200 ; 66376,201317,746415
AAAAAGCAGAAATCTTTGAG : 416040 ; 377108
AAAAATTGAAAATTTTGACG : 468060 ; 529847,1059379

…

TTTAAATTTTTTATTTTTTT : -630060 ; 562855
TTTAACAAAAGAATTAATTG : 870500 ; 532942
TTTAACTATAAAAAGATTTT : 722240 ; 1010252
TTTAACTTCGTAATCGTATT : 1408200 ; 228056
TTTAAGAGTTGAAATGTTTG : 832600 ; 490446
TTTAAGCTGTCTTTTTCAAA : -1371160 ; 1124993

…
```

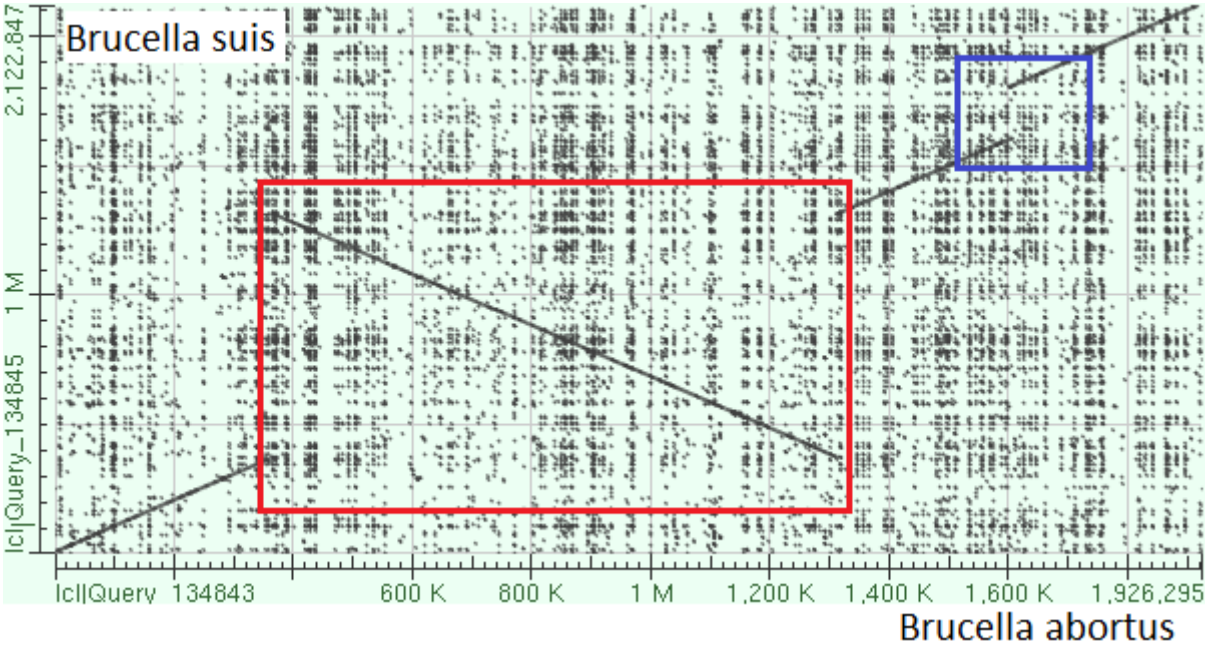
Следует отметить, что наличие вхождений с инверсиями в файлах отмечается отрицательным значением индекса, т.е. пройдя от начала генома количество нуклеотидов, равное абсолютному значению индекса вхождения, а дальше считывая последовательно нуклеотиды влево, если индекс отрицательный, и вправо, если положительный, мы получим подстроку, вхождения которой мы рассматриваем.

Так же в файле информация о вхождении подстрок лексикографически упорядочена, поэтому поиск индексов вхождения подстроки или факта наличия вхождения подстроки одного генома в другой выполняется с помощью бинарного поиска, что дает сложность $O(log\ n)$, что довольно полезно в нашем случае - случае работы с большими данными.

По итогу работы нашего алгоритма выравнивания создается файл, в котором в упорядоченном виде записываются все вхождения подстрок определенной длины одной из последовательностей в другую. На этом выравнивание геномов заканчивается.

Визуализация данных

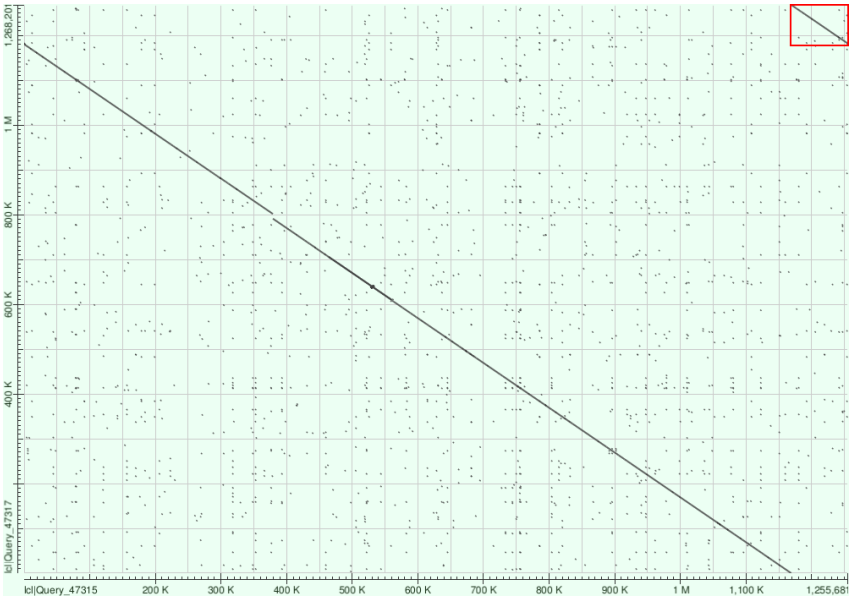
Для визуального представления результатов выравнивания геномов можно использовать аналоги карт локального сходства[1], что дальше и будет использовано для объяснения результатов и алгоритмов анализа результата выравнивания.



Оси графика можно представить в виде двух последовательностей всех индексов нуклеотидов - по оси на геном. Далее маленьким отрезком черного цвета отмечается наличие вхождения подстроки в обе последовательности, т.е. выбирается на графике место, где входит эта подстрока в один геном, т.е. выбирается координата по одной оси, далее аналогичным образом, но для другой последовательности, ищется место вхождения в нее, т.е. координата по второй оси. Так происходит для двух концов подстроки - начала и конца. По этим двум точкам строится отрезок. Каждый из отрезков получается достаточно маленьким по сравнению с масштабом графика, поэтому будет считать, что любой такой отрезок - просто точечное вхождение в обе последовательности.

Обсудим, как выглядят те или иные геномные перестройки на графике локального сходства: Красным прямоугольником выделена часть с инверсией в одном из рассматриваемых геномов. Синим прямоугольником обозначена инсерция или делеция - зависит от того, по какому геному мы смотрим: если считать геном, имеющий горизонтальную ось, главным, то синим прямоугольником выделена инсерция фрагмента во второй, вертикальный геном; если взять за главный геном второй, то синий прямоугольник выделяет делецию в геноме.

Следует взять во внимание, что в некоторых случаях можно увидеть смещение совпадений геномов по какой-либо из осей графика, как на этой карте сходства:



Такое смещение возникает из-за того, что сам геном - закольцованная структура, у него нет конца или начала, поэтому при его секвенировании может произойти вот такое смещение последовательности нуклеотидов.

Несмотря на то, что это очевидно, хочется отметить, что отрезок, соединяющий начало графика и точку, имеющую максимальные по обеим осям координаты, обозначает практически полное сходство двух геномов.

Анализ выравнивания

Получив схожие отрезки геномов, можно приступать к их анализу. Нам необходимо было выделить ключевые события-мутации, произошедшие с одним геномом, по сравнению с другим. Как было сказано ранее, если две последовательности схожи, то карта сходства будет представлять из себя прямую $y = x$. Имея в распоряжении итоговый вариант, мы восстанавливали события с конца, таким образом на выходе у нас должна была получиться как можно более ровная прямая и набор геномных мутаций.

Полный алгоритм анализа можно разделить на несколько этапов:

1. Преобразование входных данных и построение новых отрезков
2. Определение сдвига и инверсий
3. Определение вставок и инверсий
4. Определение транслокаций

Чтобы наглядно продемонстрировать каждый этап, будут приведены карты сходства второй хромосомы бактерии brucella abortus штамма 104M и второй хромосомы бактерии Brucella suis штамма Bs 143 CITAs.

Как было сказано ранее, входные данные анализа (они же – результат работы выравнивания) достаточно объёмны и приводить их не имеет смысла – они лишь содержат описание отрезков и их положений.

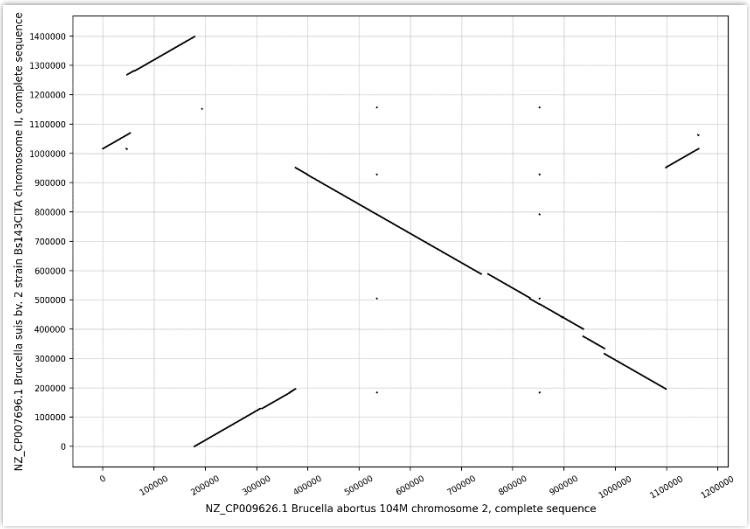
Для анализа, в отличии от выравнивания, был выбран язык Python, который позволяет эффективнее вносить изменения в алгоритм, что было очень важно в процессе разработки, поскольку используемые методики полностью не были освещены в существующих научных работах.

Преобразование входных данных и построение новых отрезков:

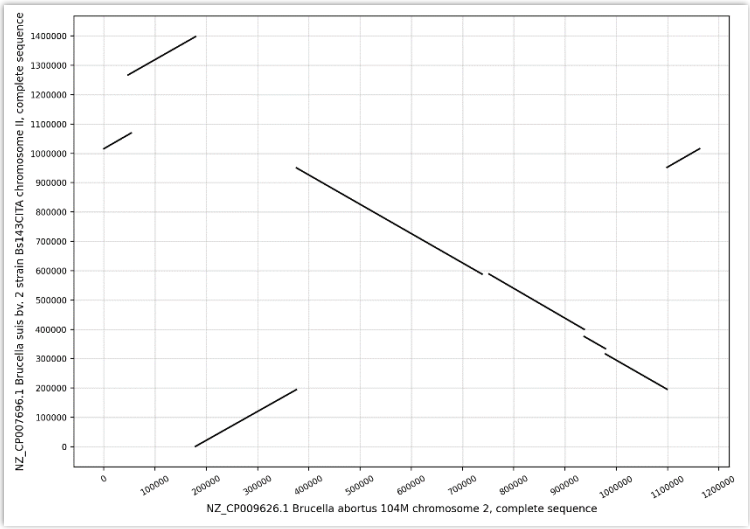
В начале, после выравнивания, все сегменты снова разбиваются на точки. Это было сделано по двум причинам. Во-первых, для разделения выравнивания и анализа, как двух отдельных алгоритмов, а соответственно их универсальности. Во-вторых, из-за того же разделения, на протяжении почти всей работы в качестве входных данных для анализа использовался алгоритм BWA, вывод которого также было удобно переводить в точки.

Burrows-**W**heeler **A**ligner – это программный пакет, позволяющие осуществить выравнивание геномов с помощью алгоритма BWT. **B**urrows-**W**heeler **T**ransform – это один алгоритм сжатия данных, активно применяющийся в программировании. BWA использовался в анализе, как временная замена собственного алгоритма, для получения результатов выравнивания, которые нужны для запуска, проверки и тестирования анализа.

Отрезки заново строятся по точкам: каждая следующая точка добавляется к предыдущему отрезку тогда, когда она находится достаточно близко к нему. Иначе она является началом нового отрезка. Чтобы получить чистую картинку и убрать шум, был использован порог, отрезки с длиной ниже которого, отсеиваются. Сравнение графика **без** порога по размеру и **с** порогом представлено далее. Карта сходства справа является результатом работы программы на данном этапе.



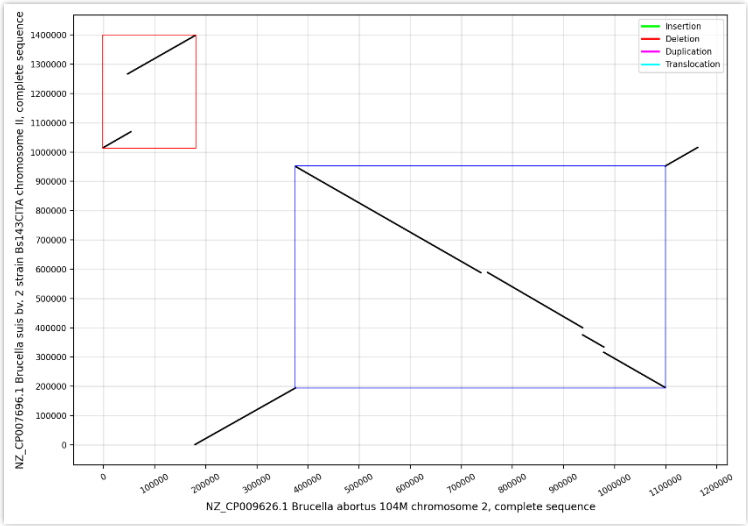
Карта сходства **без** порога по размеру отрезка



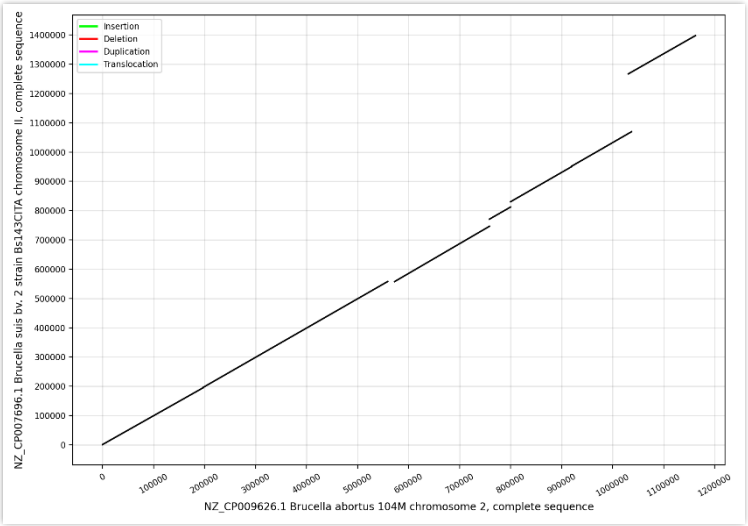
Карта сходства **с** порогом по размеру отрезка

Определение сдвигов и инверсий

Для определения сдвигов и инверсий мы использовали перебор, чтобы абстрагироваться от констант и унифицировать алгоритм. Поскольку количество отрезков в среднем не превышает 30, а теоретический максимум при разумных входных данных может составить не больше 100-500 отрезков, мы могли использовать асимптотику вплоть до $O(n^3)$. Алгоритм перебора был построен таким образом, что он сразу производил подсчёт как сдвигов, так и инверсий, таким образом на его выходе мы сразу получали карту сходства, которая наилучшим образом описывалась прямой. Мы протестировали различные метрики для перебора и пришли к выводу, что разница координат точки, возведённая в квадрат $(y - x)^2$, работает наилучшим образом для бóльшего количества пар геномов. По сути, такая метрика показывает **расстояние по оси Y** от данной точки до прямой $y = x$. Проиллюстрируем результат на этом этапе:



Результат первого этапа
входные данные второго этапа



Результат второго этапа

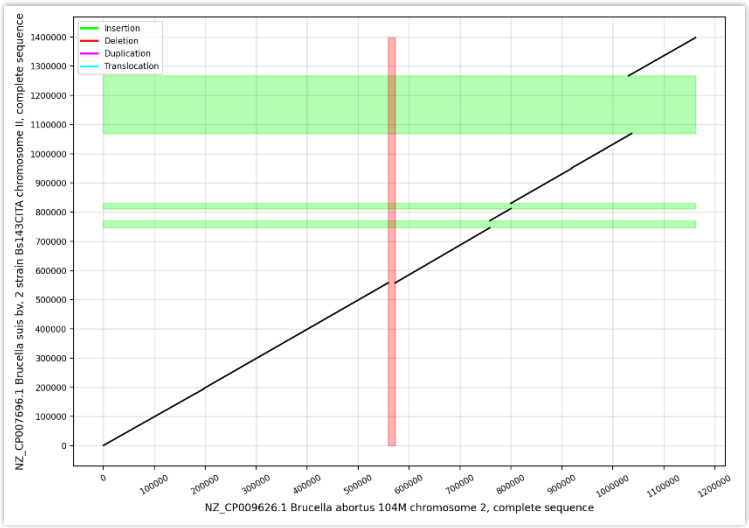
Заметим следующие изменения:

- Очевидно, красный отрезок является концом генома, который оказался в начале из-за **сдвига** области секвенирования. Программа это обнаружила и переместила его в конец так, что на правом графике его даже не видно – настолько хорошо он вписался в предыдущий отрезок.
- Отрезок, выделенный синим, был подвержен **инверсии**. Таким образом повернув его ещё раз, мы её компенсируем, и он также окажется ровно в створе предыдущего и следующего отрезков.

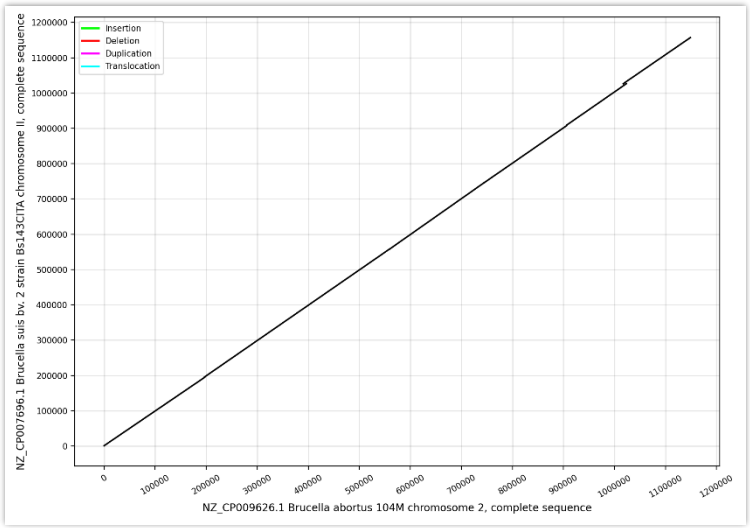
Определение вставок и делеций

Посмотрев на вывод второго этапа, можно понять, что определение вставок и делеций осуществить очень легко:

Спроецируем каждый отрезок на оси X и Y. Тогда каждый промежуток на оси X будет означать делецию, а промежуток на оси Y – вставку. Промежутки между отрезками создают полосы (см. картинку), так что затем остаётся лишь вырезать эти полосы. На следующей карте сходства изображена только делеция, поскольку в данной паре геномов нет больших вставок, однако в других случаях они имеют место быть.



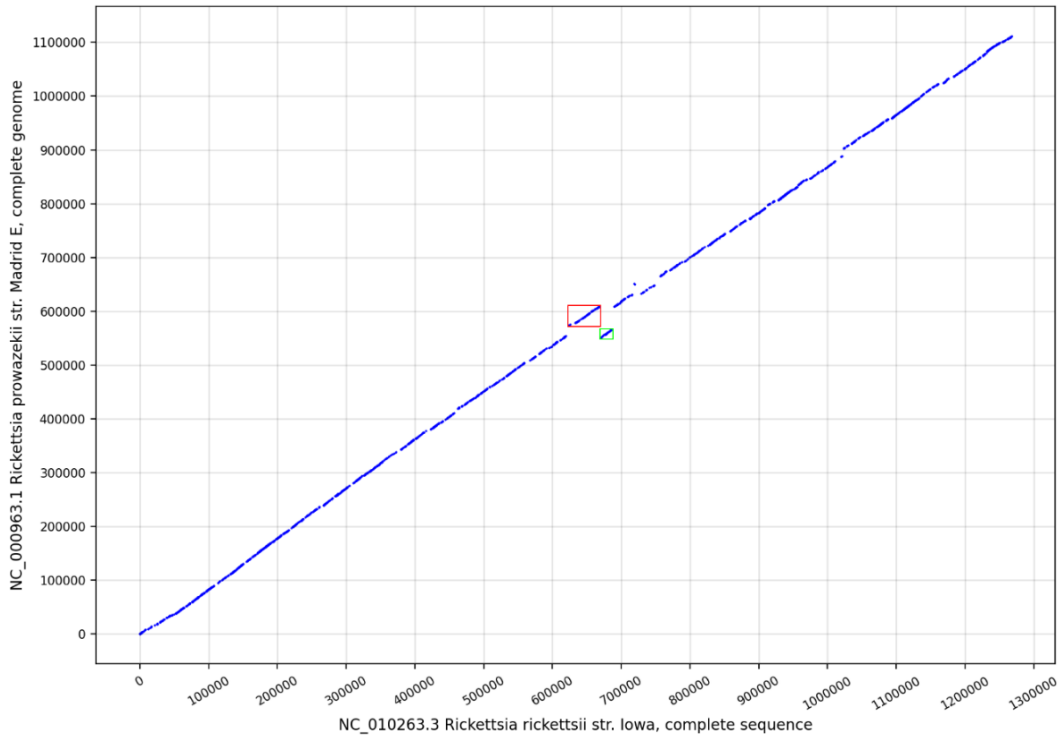
Красной полосой отмечена делеция



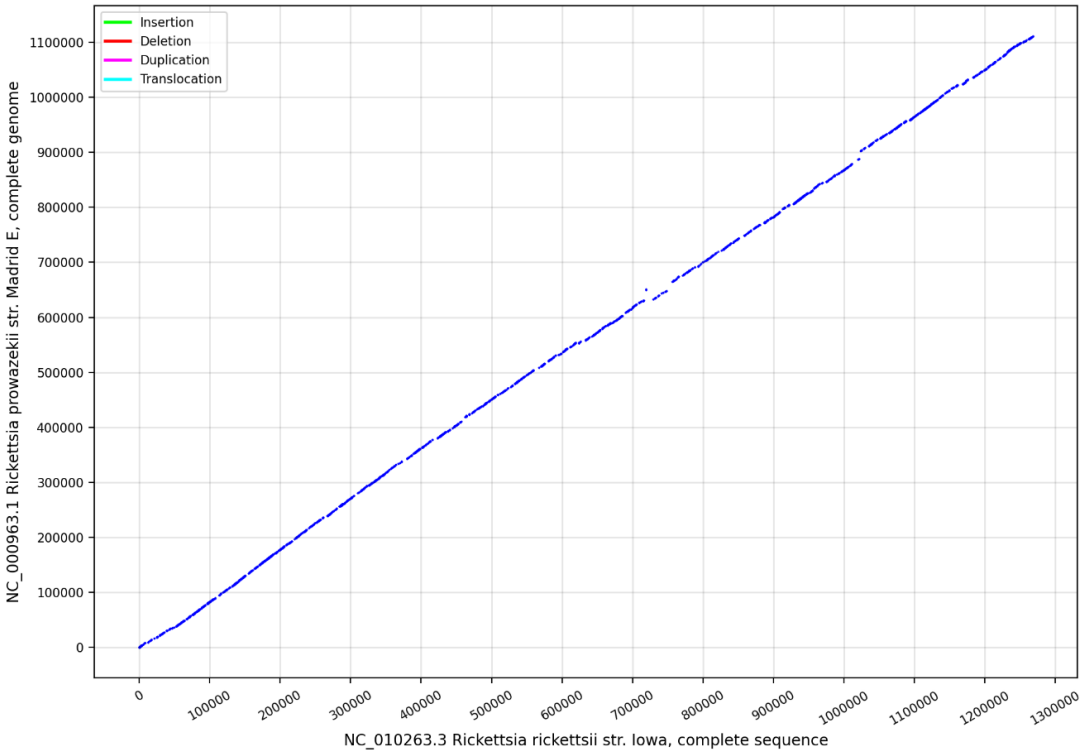
Результат третьего этапа
(после удаления делеции)

Определение транслокаций

Поскольку в выравнивании пары геномов, которое приводилось выше, отсутствует транслокация, для демонстрации следующего этапа используем геномы бактерий Rickettsia rickettsii str. Iowa и Rickettsia prowazekii str. Madrid E, карта сходства которых после предыдущих этапов выглядит следующим образом:

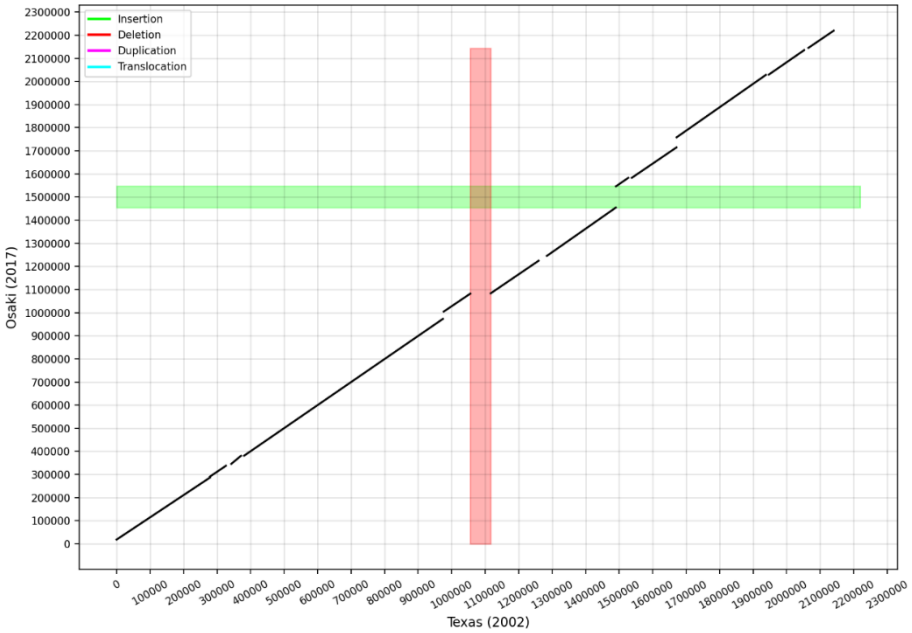


Здесь уже представлены не обычные отрезки, а точки, которые эти отрезки составляют. Можно заметить, что отрезки, выделенные красным и зелёным цветом расположены в обратном порядке и должны быть поменяны местами - в данном случае это транслокация. Результат обработки транслокаций:



Другие примеры

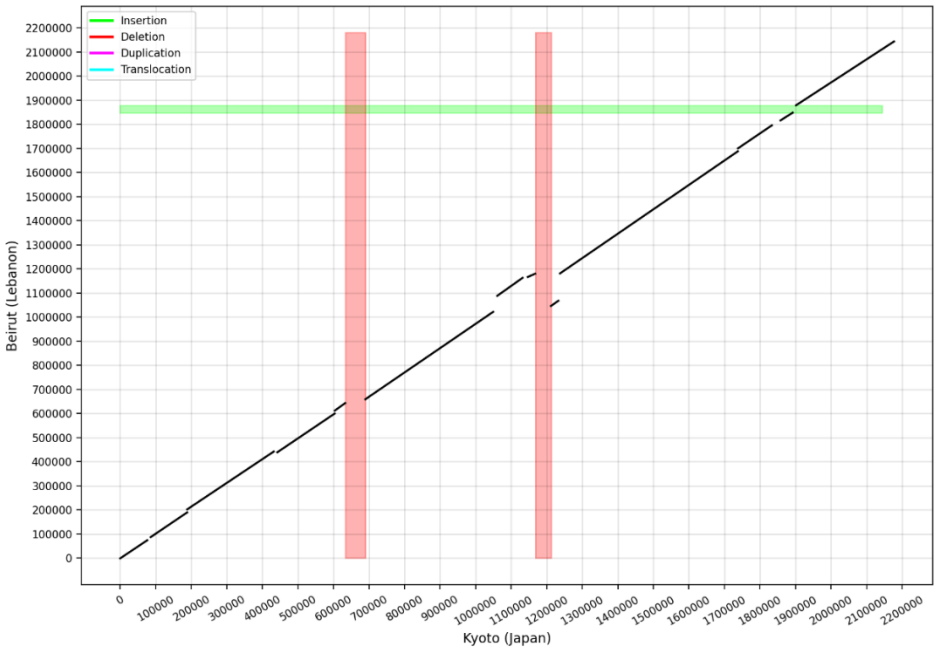
Посмотрим на сравнение штаммов одного из сильнейших[2] пневмонии, который может расти и размножаться в бескислородных условиях – **пневмонийного стрептококка** образцов HYPERLINK "https://www.ncbi.nlm.nih.gov/nuccore/NZ_CP053210.1"2002 года (Техас, США) и HYPERLINK "https://www.ncbi.nlm.nih.gov/nuccore/NZ_CP050175.1"2017 года (Осаки,



Япония):2017 года (Осаки, Япония):2017 года (Осаки, Япония):

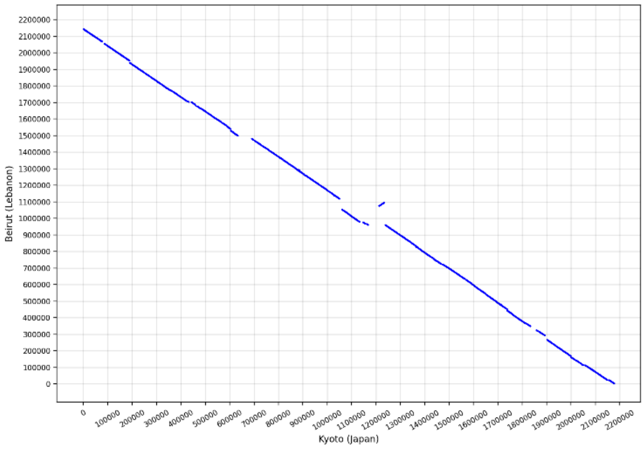
Экспериментально известно, что за эти 15 лет вирус мутировал. На карте сходства мы видим большую область **(инсерцию)**, которая присутствует на штамме 2002 года, но отсутствует в 2017 году - скорее всего она и отвечает за устойчивость микроорганизма.

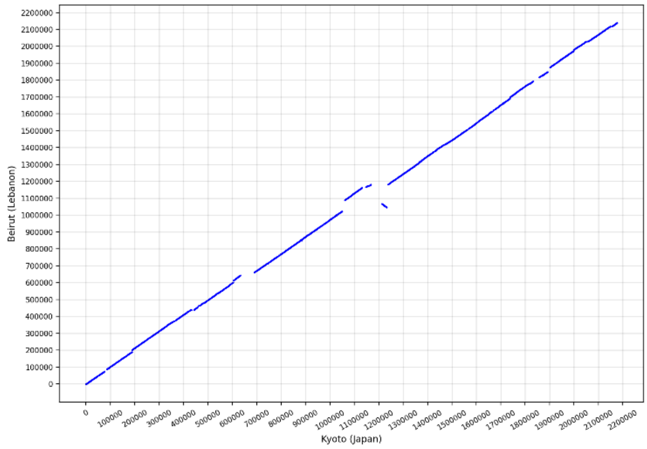
Можно также посмотреть на сравнение штаммов 2017 года в городах Киото (Япония) и Бейрут (Ливан):

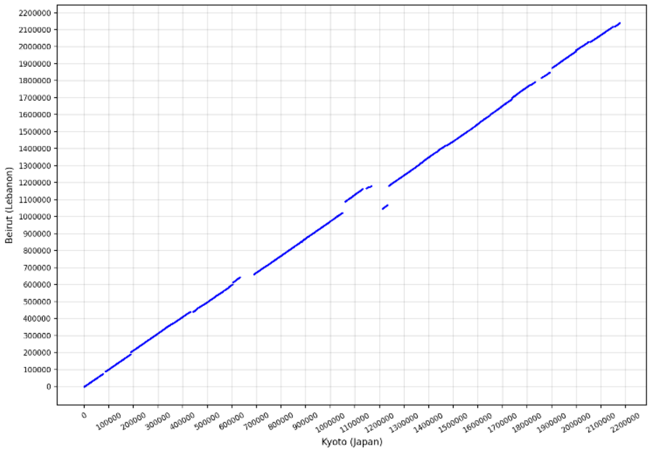


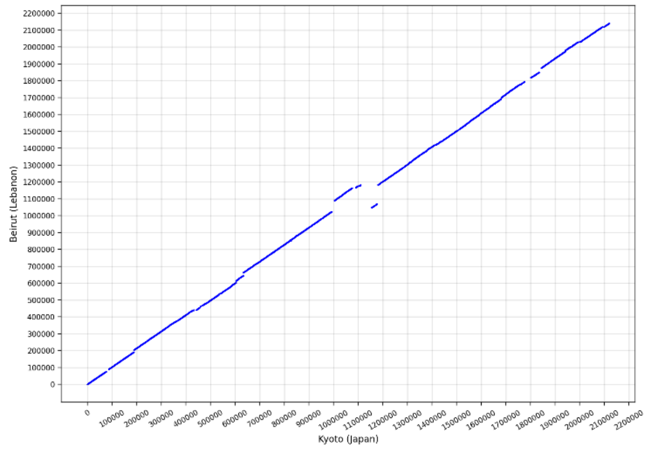
Однако здесь можно заметить куда меньше различий – одна инверсия и небольшие вставки, которые, скорее всего, не сильно влияют на разницу бактерий.

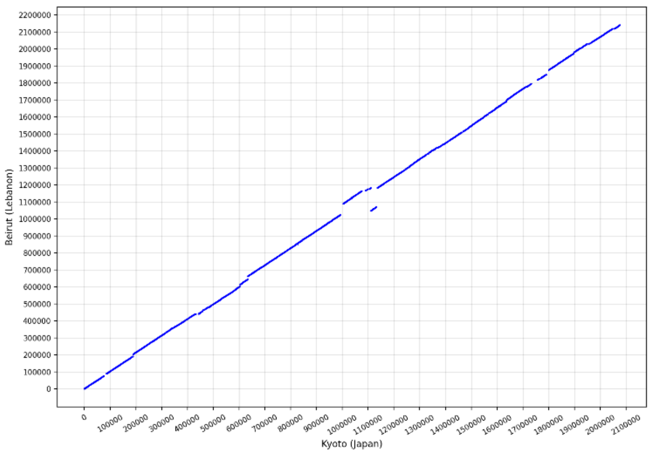
Также ниже представлен пример визуализации последовательности эволюционных мутаций, произошедших с этой парой геномов (кроме первого шага – смещения из-за разности областей секвенирования):

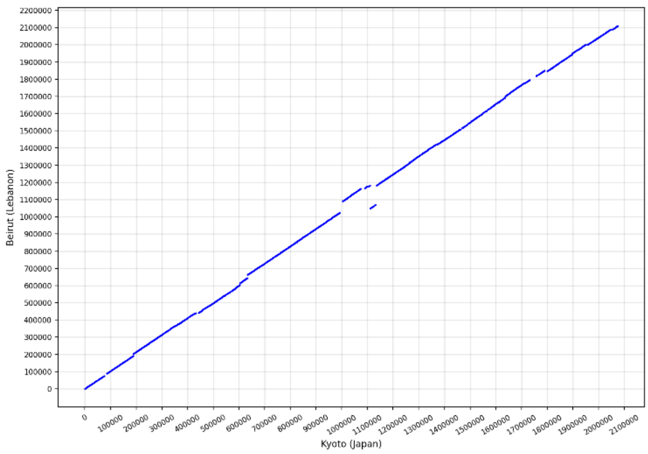


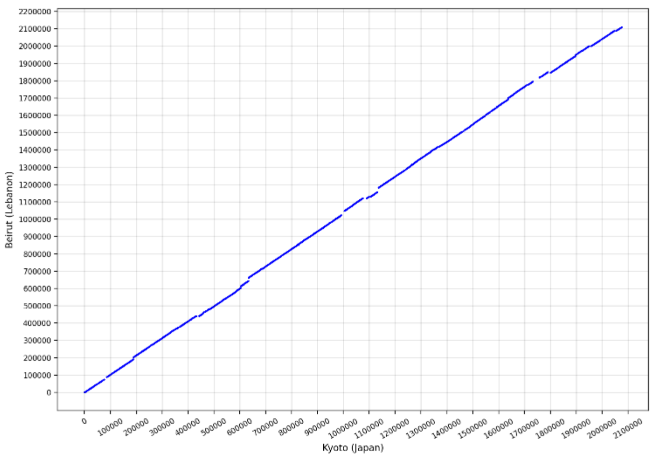












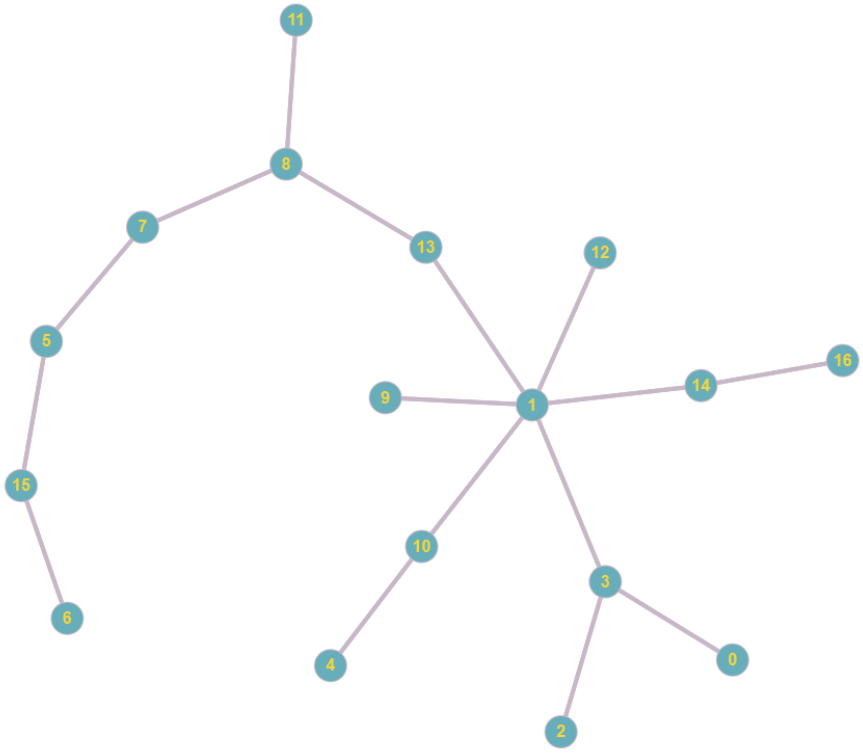
Структуры родственности

Под “структурами родственности” мы подразумеваем родственное дерево. Мы можем его составить, поскольку у нас уже есть программа, позволяющая в какой-то мере степень родства двух геномов. Рассмотрим подробнее вывод анализа, который до этого момента так и не был представлен:

После завершения работы над парой геномов, мы получаем в текстовом виде список самых больших мутаций, которые связывают эти две последовательности. Приведём пример:

```
Rotation from 0 (Query) to 2'219'730 (Query)
Insertion of 657'311-762'698 (Ref)
Deletion of 1'076'530-1'156'940 (Query)
Deletion of 673'190-738'740 (Query)
```

В этом, наиболее приближенным к привычному человеку виду, тексте описаны: поворот, вставка и две делеции вместе с координатами. По такому файлу можно составить какую-либо простую метрику, которая позволит построить граф, у которого между каждой парой вершин будет ребро с весом равной этой метрике. Затем этот граф преобразуется в дерево. Ниже приставлено дерево, построенный таким способом, который состоит из 17 различных штаммов **пневмонийного стрептококка**:



Сейчас ведётся работа в данном направлении, поэтому здесь мы не приводим детальное описания графика и процесса его построения.

В будущем мы планируем полностью автоматизировать этот процесс так, чтобы все три этапа работы выполнялись последовательно без калибровки и участия человека.

ВЫВОДЫ

В процессе исследования крупных геномных перестроек внутри пар бактерий одного вида было обнаружено, что фактически все геномы в рассматриваемых парах являются составными относительно друг друга, т.е. из одного генома путем конечного числа таких преобразований, как инверсия, дупликация фрагмента и т.д., можно получить практически весь второй геном. Это позволяет с довольно большой точностью просматривать произошедшие мутационные процессы внутри пары геномов.

Также было обнаружено, что близкородственные бактерии действительно в своих геномах имеют гомологичные участки, что позволяет говорить о степени родственности количественно, пусть даже количественное определение родственности не будет говорить ни о чем, кроме наличия и размеров совпадающих участков.

В том числе можно сказать, что скорость алгоритма выравнивания получилась не очень большая (на выравнивание геномов $2Mb\ x\ 2Mb$ уходит менее 3 минут), чтобы составлять структуру родственности для порядка $10 - 20$ геномов, но из-за квадратичной скорости возрастания необходимых пар для сравнивания при добавлении очередного генома в структуру, пришлось отказаться от превышения ранее упомянутого разумного размера группы бактерий, по которым строится структура родственности.

Мы также размышляли над способами решения квадратичного возрастания необходимых сравнений при добавлении нового генома в структуру родственности, но нам в рамках практики не хватило ни времени, ни вычислительной мощности, ни, судя по всему, опыта работы с большими данными, для попытки реализации этих идей.

Можно сказать, что у нас получилось с большой точностью производить моделирование мутационных изменений в ходе эволюционного процесса, что автоматизировало то, чем раньше люди занимались вручную, а это несомненно является одним из поставленных и достигнутых результатов в рамках нашей работы.

Реализацию написанной программы и некоторых упомянутых в работе алгоритмов можно найти по адресу https://guthub.com/npanuhin/BIOCAD.

ИСТОЧНИКИ

- National Center for Biotechnology Information
- BLAST algorithm
- [1] kodomo.fbb.msu.ru
- Altschul, Stephen. (2005). BLAST Algorithm. 10.1038/npg.els.0005253.
- Na, Joong & Kim, Hyunjoon & Min, Seunghwan & Park, Heejin & Lecroq, Thierry & Leonard, Martine & Mouchard, Laurent & Park, Kunsoo. (2016). FM-index of Alignment with Gaps. Theoretical Computer Science. 710. 10.1016/j.tcs.2017.02.020.
- Gambin, Anna & Lasota, Slawomir & Startek, Michal & Sykulski, Maciej & Noé, Laurent & Kucherov, Gregory. (2011). Subset seed extension to Protein BLAST. BIOINFORMATICS 2011 - Proceedings of the International Conference on Bioinformatics Models, Methods and Algorithms. 149-158. 10.5220/0003147601490158.
- Morgulis, Aleksandr & Gertz, E & Schaffer, Alejandro & Agarwala, Richa. (2006). A Fast and Symmetric DUST Implementation to Mask Low-Complexity DNA Sequences. Journal of computational biology : a journal of computational molecular cell biology. 13. 1028-40. 10.1089/cmb.2006.13.1028.
- [2] “Despite vaccines, Streptococcus pneumoniae kills more than a million people yearly.” (2017)

БЛАГОДАРНОСТИ

Департаменту вычислительной биологии BIOCAD