Всероссийский конкурс исследовательских и проектных работ школьников «Высший пилотаж»

«Программный пакет RNAsselem на языке Python для анализа вторичных структур РНК.»

Проект

Направление «Computer Science»

Автор: Казанов Федор Маратович,
учащийся 11класса, ЧУОО СШ “Знайка”, г. Москва

Руководитель проекта: Пономарев Геннадий Владимирович, научный сотрудник Сколтех, ИОГен РАН.

2025 г.

Аннотация

Молекулы РНК играют важную роль в различных молекулярных процессах внутри клеток. Вторичная структура молекулы РНК, а именно образование дополнительных водородных связей между малыми молекулами, составляющими молекулу РНК, часто отвечает за выполнение ключевых функций в жизненном цикле клеток и вирусов. Для изучения роли вторичной структуры РНК важно использовать вычислительные инструменты для анализа различных элементов вторичной структуры, включая шпильки, внутренние петли, выпячивания, многоразветвленные петли, стебли и внешние петли. В данном проекте я разработал программный пакет на языке Python для анализа элементов вторичной структуры РНК, который позволяет распознавать основные паттерны вторичной структуры, генерировать описательную статистику для этих структурных элементов и получать их основные характеристики. Разработанный пакет был применен для анализа полных вирусных геномов. Программный пакет, названный RNAsselem, размещен для установки и использования на серверах Github и PyPi.

Актуальность проекта

Основными компонентами клетки являются молекулы ДНК, РНК и белков. Молекула РНК, как и ДНК, является полимером - молекулой, состоящей из однотипных малых молекул, называемых нуклеотидами [1]. Существуют 4 вида нуклеотидов: аденин, тимин, цитозин и гуанин, которые обозначаются соответственно буквами A, T, C, G. Комплементарными нуклеотидами являются пары аденина с тимином (A – T) и пары цитозина с гуанином (C – G). Комплементарные нуклеотиды образуют водородные связи [2]. В отличии от двухцепочечной молекулы ДНК, молекула РНК может образовывать различные трехмерные структуры, за счет спаривания нуклеотидов, находящихся в произвольных позициях. Информацию о спаренности нуклеотидов для молекулы РНК называют ее вторичной структурой [3]. Многие типы молекул РНК, такие как рибосомальная РНК и транспортная РНК, имеют определённую вторичную структуру, важную для выполнения их функций [4]. Также вторичная структура РНК имеет большое значение у РНК-вирусов, геномы которых являются длинной молекулой РНК, состоящей из тысяч нуклеотидов [5]. Изучению вторичной структуры РНК посвящается множество исследований, а информацию о вторичной структуре, то есть информацию о спаренности нуклеотидов, принято хранить в файлах специального формата [6]. Вторичная структура может быть логически разбита на некоторые однотипные структурные элементы, такие как стебли, внутренние и внешние петли, шпильки, многоразветвлённые петли, выпячивания (Рис 1) [7]. Последние исследования показывают важность изучения элементов вторичных структур РНК, так как именно отдельные элементы вторичной структуры могут играть значительную функциональную роль. Известными примерами являются RRE элемент (Rev response element), ответственный за селективный экспорт молекул РНК из ядра клетки [8] и FSE элемент (frameshifting stimulation element) вируса SARS-CoV2, выполняющего программный сдвиг рибосомы [9]. Так как роль вторичной структуры РНК у РНК-

вирусов была открыта относительно недавно, на данный момент практически не существует специализированных программных пакетов для анализа элементов вторичной структуры РНК. Поэтому разработка программного пакета, который бы мог выполнять распознавание элементов вторичной структуры на основе входных файлов с информацией о спаренности нуклеотидов, а также выполнять их анализ и подсчитывать описательную статистику, является актуальной задачей.

Литература:

1. Caprara, M. G. & Nilsen, T. W. RNA: Versatility in form and function. Nat. Struct. Biol. 7, 831–833 (2000).
2.Strobel, E. J., Watters, K. E., Loughrey, D. & Lucks, J. B. RNA systems biology: Uniting functional discoveries and structural tools to understand global roles of RNAs. Curr. Opin. Biotechnol. 39, 182–191 (2016).
3. Ganser, L. R., Kelly, M. L., Herschlag, D. & Al-Hashimi, H. M. The roles of structural dynamics in the cellular functions of RNAs. Nat. Rev. Mol. Cell Biol. 20, 474–489 (2019).
4. Spitale, R. C. & Incarnato, D. Probing the dynamic RNA structurome and its functions. Nat. Rev. Genet. 24, 178–196 (2023).
5. Boerneke, M. A., Ehrhardt, J. E. & Weeks, K. M. Physical and functional analysis of viral RNA genomes by SHAPE. Annu. Rev. Virol. 6, 93–117 (2019).
6. Nawrocki, E. & Eddy, S. RNA Secondary Structures: WUSS Notation, INFERNAL User's Guide. 107–108 http://eddylab.org/inferna l/Userguide.pdf (2023).
7. Mathews, D. H. RNA secondary structure analysis using RNAstructure. Curr. Protoc. Bioinform. https://doi.org/10.1002/0471250 953.bi1206s46 (2014).
8. Fang, X. et al. XAn unusual topological structure of the HIV-1 rev response element. Cell 155, 594 (2013).
9. Hill, C. H. & Brierley, I. Structural and functional insights into viral programmed ribosomal frameshifting. Annu. Rev. Virol. 10 (2023).

Цели проекта

Задачей проекта была разработка программного пакета, который выполнял бы следующие задачи:

- Конвертацию вторичной структуры из форматов dot-bracket или CT в формат WUSS, где каждый нуклеотид классифицируется как часть стебля или одного из пяти типов петлевых областей: шпилька (hairpin loop), внутренняя петля (internal loop), выпячивание (bulge loop), многоразветвлённая петля (multifurcation loop) или внешняя петля (external loop);
- Сбор классифицированных нуклеотидов в отдельные элементы вторичной структуры;

- Получение информации об элементе вторичной структуры для конкретной позиции в последовательности молекулы РНК;
- Генерацию списка структурных элементов определённого типа с указанием их позиций и размеров, включая шпильки, внутренние петли, выпуклости и разветвления;
- Вычисление статистики покрытия генома различными типами структурных элементов;
- Вычисление статистики для структурных элементов определённого типа, включая частоту их встречаемости в геноме, а также расчёт среднего, медианы, стандартного отклонения размеров элементов и их общей длины в геноме.
- Вычисление статистики для спаренных и неспаренных нуклеотидов;

Методы работы

Типы вторичных структур

Существуют различные типы элементов вторичных структур:

1) stem (стебель) - структурный элемент, состоящий из двух спаренных участков молекулы РНК, расположенных удаленно друг от друга в пределах цепи РНК. В данной работе стебель, прерывающийся внутренними петлями и выпячиваниями, считается одним элементом, а стебли, прерывающиеся многоразветвленными петлями, считаются разными элементами.
2) internal loop (внутренняя петля) - это участок вторичной структуры РНК, который содержит неспаренные нуклеотиды на обеих цепях стебля. Внутренние петли ограничены парами спаренных нуклеотидов, которые замыкают петлю с обеих сторон.
3) bulge loop (выпячивание) - это участок вторичной структуры РНК, в котором неспаренные нуклеотиды присутствуют только на одной из цепей стебля. Выпячивание ограничено спаренными нуклеотидами, которые замыкают петлю с обеих сторон.
4) multifurcation loop (многоразветвленная петля) - это участок вторичной структуры РНК, представляющий собой области кольца, соединяющего между собой три или более стебля. Стебли не являются частью многоразветвленной петли.
5) hairpin loop (петля шпильки) – это участок между двумя комплементарными областями стебля, состоящий из неспаренных нуклеотидов, который образует “шапочку” над стеблем
6) external loop (внешняя петля) - участок неспаренных нуклеотидов, не относящийся к другим структурным элементам.

Все эти типы представлены на рисунке 1.

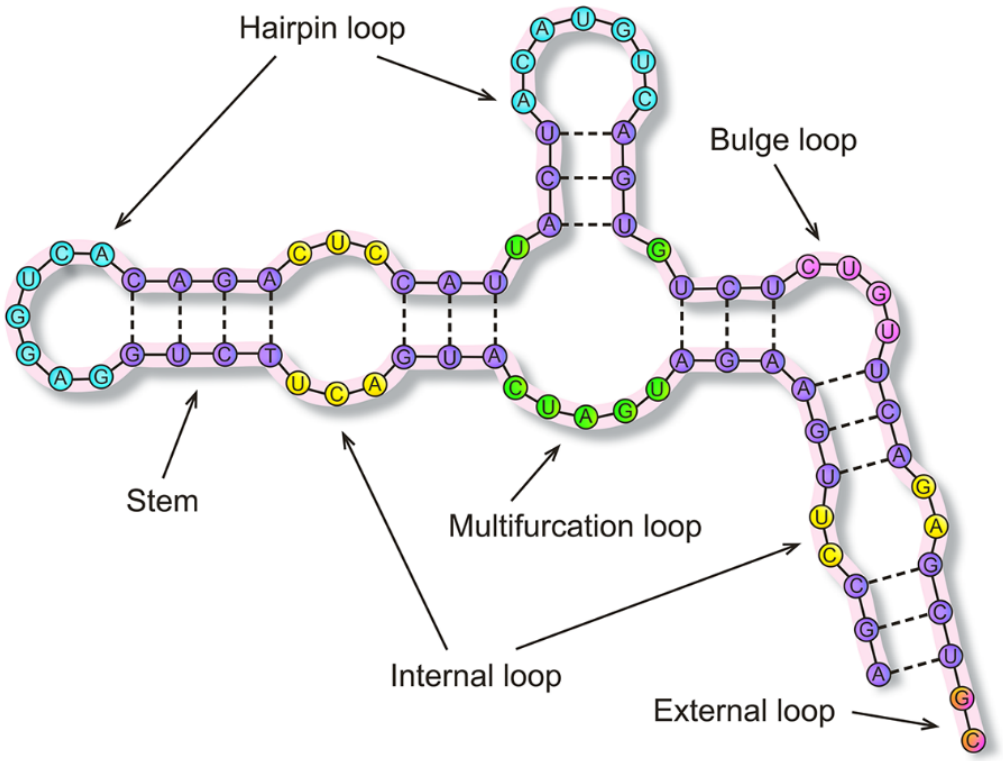


Рисунок 1. Виды элементов вторичной структуры.

Входные данные

Входными данными для программного пакета являлись файлы с информацией о вторичной структуре в трех различных форматах.

- Файл формата dot-bracket. Файл такого формата устроен следующим образом: комплементарные нуклеотиды обозначаются открытыми и закрытыми скобками в соответствующих позициях. Неспаренные нуклеотиды обозначаются точками (Рис 2).

```
>Example
AGCCUUGAAGAUGAUCAUGACUTCUGGAGGUCACAGACUCCAUUACUACAUGUCAGUGUCUCUGUUCAGAGCUGC
(((..((((((.....(((...(((((.......)))))...)))).(((.......))).)))....)))..)))..
```

Рисунок 2. Файл формата dot-bracket (пример для вторичной структуры, изображенной на рис.1).

- Файл формата connectivity table (CT). В файле данного формата все позиции молекулы РНК пронумерованы, и номер спаренного нуклеотида указывается в отдельном столбце (Рис 3).

```
75 Example
  1 A   0   2 73   1  |  16 C 15 17   0 16  |  31 U 30 32   0 31  |  46 C 45 47 56 46  |  61 U 60 62   9 61
  2 G   1   3 72   2  |  17 A 16 18 43 17  |  32 C 31 33   0 32  |  47 U 46 48 55 47  |  62 C 61 63   0 62
  3 C   2   4 71   3  |  18 U 17 19 42 18  |  33 A 32 34   0 33  |  48 A 47 49   0 48  |  63 U 62 64   0 63
  4 C   3   5  0   4  |  19 G 18 20 41 19  |  34 C 33 35 26 34  |  49 C 48 50   0 49  |  64 G 63 65   0 64
  5 U   4   6  0   5  |  20 A 19 21   0 20  |  35 A 34 36 25 35  |  50 A 49 51   0 50  |  65 U 64 66   0 65
  6 U   5   7 68   6  |  21 C 20 22   0 21  |  36 G 35 37 24 36  |  51 U 50 52   0 51  |  66 U 65 67   8 66
  7 G   6   8 67   7  |  22 U 21 23   0 22  |  37 A 36 38 23 37  |  52 G 51 53   0 52  |  67 C 66 68   7 67
  8 A   7   9 66   8  |  23 T 22 24 37 23  |  38 C 37 39   0 38  |  53 U 52 54   0 53  |  68 A 67 69   6 68
  9 A   8 10 61   9  |  24 C 23 25 36 24  |  39 U 38 40   0 39  |  54 C 53 55   0 54  |  69 G 68 70   0 69
 10 G   9 11 60 10  |  25 U 24 26 35 25  |  40 C 39 41   0 40  |  55 A 54 56 47 55  |  70 A 69 71   0 70
 11 A 10 12 59 11  |  26 G 25 27 34 26  |  41 C 40 42 19 41  |  56 G 55 57 46 56  |  71 G 70 72   3 71
 12 U 11 13   0 12  |  27 G 26 28   0 27  |  42 A 41 43 18 42  |  57 U 56 58 45 57  |  72 C 71 73   2 72
 13 G 12 14   0 13  |  28 A 27 29   0 28  |  43 U 42 44 17 43  |  58 G 57 59   0 58  |  73 U 72 74   1 73
 14 A 13 15   0 14  |  29 G 28 30   0 29  |  44 U 43 45   0 44  |  59 U 58 60 11 59  |  74 G 73 75   0 74
 15 U 14 16   0 15  |  30 G 29 31   0 30  |  45 A 44 46 57 45  |  60 C 59 61 10 60  |  75 C 74 76   0 75
```

Рисунок 3. Файл формата CT (пример для вторичной структуры, изображенной на рис.1)..

- Файл формата CT-WUSS. Данный файл аналогичен файлу типа CT, за исключением дополнительного столбца, в котором приведены аннотации элементов вторичных структур. Нуклеотиды, принадлежащие к структурным элементам hairpin loop, internal loop, bulge loop, multifurcation loop, external loop обозначаются соответственно символами “_”, “-”, “-”, “,”, “:”. Однако stem обозначатся различными символами. Два комплементарных нуклеотида стебля обозначаются различными символами, один из них – открывающейся скобкой, другой – закрывающейся. Нуклеотиды, принадлежащие стеблю, имеют различные обозначения в зависимости от уровня вложенности: Первый уровень вложенности (внешний) обозначается символом “<” или “>”, второй – символом “(“ или “)”, третий - символом “[“ или “]”, четвертый - символом “{“ или “}”.

```
75 Example
  1 A   0   2 73   1 (  |  16 C 15 17   0 16 ,  |  31 U 30 32   0 31 _  |  46 C 45 47 56 46 <  |  61 U 60 62   9 61 )
  2 G   1   3 72   2 (  |  17 A 16 18 43 17 <  |  32 C 31 33   0 32 _  |  47 U 46 48 55 47 <  |  62 C 61 63   0 62 -
  3 C   2   4 71   3 (  |  18 U 17 19 42 18 <  |  33 A 32 34   0 33 _  |  48 A 47 49   0 48 _  |  63 U 62 64   0 63 -
  4 C   3   5  0   4 -  |  19 G 18 20 41 19 <  |  34 C 33 35 26 34 >  |  49 C 48 50   0 49 _  |  64 G 63 65   0 64 -
  5 U   4   6  0   5 -  |  20 A 19 21   0 20 -  |  35 A 34 36 25 35 >  |  50 A 49 51   0 50 _  |  65 U 64 66   0 65 -
  6 U   5   7 68   6 (  |  21 C 20 22   0 21 -  |  36 G 35 37 24 36 >  |  51 U 50 52   0 51 _  |  66 U 65 67   8 66 )
  7 G   6   8 67   7 (  |  22 U 21 23   0 22 -  |  37 A 36 38 23 37 >  |  52 G 51 53   0 52 _  |  67 C 66 68   7 67 )
  8 A   7   9 66   8 (  |  23 T 22 24 37 23 <  |  38 C 37 39   0 38 -  |  53 U 52 54   0 53 _  |  68 A 67 69   6 68 )
  9 A   8 10 61   9 (  |  24 C 23 25 36 24 <  |  39 U 38 40   0 39 -  |  54 C 53 55   0 54 _  |  69 G 68 70   0 69 -
 10 G   9 11 60 10 (  |  25 U 24 26 35 25 <  |  40 C 39 41   0 40 -  |  55 A 54 56 47 55 >  |  70 A 69 71   0 70 -
 11 A 10 12 59 11 (  |  26 G 25 27 34 26 <  |  41 C 40 42 19 41 >  |  56 G 55 57 46 56 >  |  71 G 70 72   3 71 )
 12 U 11 13   0 12 ,  |  27 G 26 28   0 27 _  |  42 A 41 43 18 42 >  |  57 U 56 58 45 57 >  |  72 C 71 73   2 72 )
 13 G 12 14   0 13 ,  |  28 A 27 29   0 28 _  |  43 U 42 44 17 43 >  |  58 G 57 59   0 58 ,  |  73 U 72 74   1 73 )
 14 A 13 15   0 14 ,  |  29 G 28 30   0 29 _  |  44 U 43 45   0 44 ,  |  59 U 58 60 11 59 )  |  74 G 73 75   0 74 :
 15 U 14 16   0 15 ,  |  30 G 29 31   0 30 _  |  45 A 44 46 57 45 <  |  60 C 59 61 10 60 )  |  75 C 74 76   0 75 :
```

Рисунок 4. Файл формата CT-WUSS.

Преобразование форматов

Для преобразования файлов вторичных структур РНК и аннотации элементов вторичной структуры были разработаны следующие программные функции:

- Была разработана функция *dotbracket2ct* для преобразования файла формата dot-bracket в файл формата CT. В качестве входных данных функция принимает два аргумента: input path – путь к dot-bracket файлу и output path – путь к CT файлу, в который будет записан результат работы данной функции. Функция устроена следующим образом:
  1) Информация из dot-bracket файла переносится в двумерный список, в котором i-й список содержит три элемента: номер нуклеотида, символ, обозначающий наличие спаренности (далее – СНС) и буква, обозначающая тип основания нуклеотида (строки пакета 1103 – 1115 на рис. 5).
  2) Создается отдельный двумерный список для нуклеотидов, принадлежащих стеблю. У каждого такого нуклеотида СНС – это “(“ или “)” (строки пакета 1117 – 1121, рис. 5).
  3) Программа находит в списке стеблей такие два подряд идущих нуклеотида, что СНС первого – это “(“, а СНС второго – это “)”. Такие два нуклеотида спарены между собой, поэтому они добавляются в словарь с парами, после чего удаляются из списка стеблей. Данный шаг выполняется, пока список стеблей не станет пустым (строки пакета 1123 – 1151, рис. 6).
  4) Так как теперь известны пары спаренных нуклеотидов, вся информация записывается в CT файл (строки пакета 1153 – 1170, рис. 6)

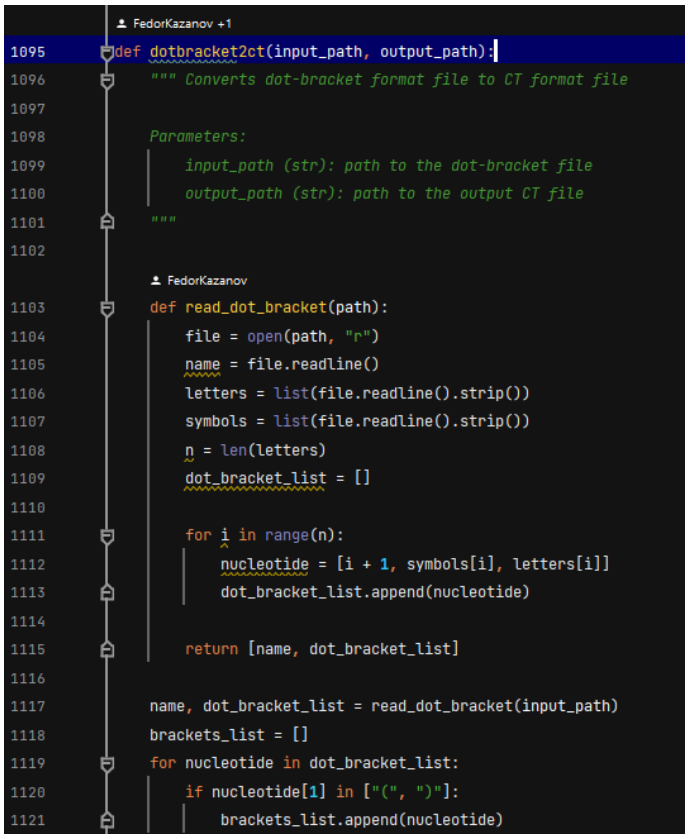


Рисунок 5. Функция dotbracket2ct , строки пакета RNAsselem 1095 – 1121.

Рисунок 6. Функция dotbracket2ct , строки пакета RNAsselem 1123 – 1170.

- Была разработана функция для преобразования файла формата CT в файл формата CT-WUSS. Алгоритм определения типов вторичных структур был адаптирован из библиотеки на языке C Easel.

Работа с элементами вторичной структуры РНК

- Предварительно была разработана функция *get_bulge_and_internal_loop_list (рис. 7)*, возвращающая список словарей отдельно как для выпячиваний, так и для внутренних петель, где i-й словарь содержит информацию об одном структурном элементе, такую как: длина элемента, массив с индексами начала и конца всех интервалов, содержащих данный элемент. В качестве входных данных функция принимает многомерный список с информацией, считанной из CT-WUSS файла. Алгоритм функции устроен следующим образом:

  1) Алгоритм линейно проходит по каждому нуклеотиду в цепи РНК.

  2) Находится пара замыкающих нуклеотидов, которые обозначаются открывающимися или закрывающимися скобками, межу которыми есть непрерывная последовательность неспаренных нуклеотидов, которые обозначаются символом “-“.

  3) Находится пара нуклеотидов, которые соответственно спарены с замыкающими нуклеотидами. Если между данной парой также есть непрерывная последовательность неспаренных нуклеотидов, которые обозначаются символом “-“, то данный элемент – это internal loop, поэтому вся информация об элементе добавляется в массив для внутренних петель, а иначе – это bulge loop, и вся информация добавляется в массив для выпячиваний.

```python
# private
FedorKazanov +1
def get_bulge_and_internal_loop_list(wuss_list):
    brackets = ["<", "(", "[", "{"]
    cl_brackets = [">", ")", "]", "}"]
    loops = ["bulge_loop", "internal_loop", "indefinite_loop"]
    length = 0
    interior_loops = []
    bulge_loops = []
    used = []
    left_index = -1

    for i in range(len(wuss_list)):
        nucleotide = wuss_list[i]
        previous_nucleotide = wuss_list[i - 1] if i != 0 else 8 * [-3]

        if nucleotide[symbol_column_number] in brackets:

            if previous_nucleotide[structure_column_number] in loops:
                right_index = nucleotide[index_column_number] - 1
                left_connection_index = wuss_list[right_index][connection_column_number] - 1
                right_connection_index = wuss_list[left_index][connection_column_number] - 1
                quantity = right_connection_index - left_connection_index - 1
                if quantity > 0:
                    total_length = length + quantity
                    intervals = [[left_index + 2, right_index], [left_connection_index + 2, right_connection_index]]
                    d = get_structure_dict(wuss_list, total_length, intervals)
                    interior_loops.append(d)
                    used.append([left_connection_index, right_connection_index])
                else:
                    dictionary = get_structure_dict(wuss_list, length, [[left_index + 2, right_index]])
                    bulge_loops.append(dictionary)
                length = 0

        elif nucleotide[symbol_column_number] in cl_brackets:

            if previous_nucleotide[structure_column_number] in loops:
                right_index = nucleotide[index_column_number] - 1

                if [left_index, right_index] not in used:
                    dictionary = get_structure_dict(wuss_list, length, [[left_index + 2, right_index]])
                    bulge_loops.append(dictionary)
                length = 0

        elif nucleotide[structure_column_number] in loops:
            if previous_nucleotide[symbol_column_number] in (brackets + cl_brackets):
                left_index = previous_nucleotide[index_column_number] - 1
            length += 1

    return [interior_loops, bulge_loops]
```

Рисунок 7. Функция get_bulge_and_internal_loop_list пакета RNAsselem.

- Была разработана функция *load_ctwuss (рис.8)*, которая загружает файл формата CT-WUSS, возвращая все данные в виде многомерного массива, который в дальнейшем можно подавать в качестве входных данных в большинство других функций. В качестве входных данных функция принимает путь к файлу CT-WUSS. Функция считывает данные из файла и вызывает внутреннюю функцию *get_bulge_and_internal_loop_list*, чтобы различить, к какой структуре относятся нуклеотиды с символом “-“. Для нуклеотидов с другими символами не требуется дополнительных действий, чтобы определить, к какой структуре относится данный нуклеотид.

```
def load_ctwuss(ctwuss_path):
    """ Loads CT-modified file with WUSS annotation in additional column.

        Parameters:
            ctwuss_path (str): path to the connectivity table (CT) file

        Returns:
            genome (list[list]): list of nucleotides with pairing info
    """
    if isinstance(ctwuss_path, list):
        return ctwuss_path

    else:
        input_file = open(ctwuss_path, "r")
        input_file.readline()
        wuss_list = []

        for line in input_file:
            nucleotide = line.split()

            for i in range(len(nucleotide)):
                symbol = nucleotide[i]
                if symbol.isdecimal():
                    nucleotide[i] = int(symbol)

            symbol = nucleotide[-1]

            if symbol == ":":
                structure_type = "external_loop"
            elif symbol == ",":
                structure_type = "multifurcation_loop"
            elif symbol == "_":
                structure_type = "hairpin_loop"
            elif symbol == "-":
                structure_type = "indefinite_loop"
            elif symbol in ["<", "(", "[", "{", ">", ")", "]", "}"]:
                structure_type = "stem"
            else:
                structure_type = "pseudoknot"

            nucleotide.append(structure_type)
            wuss_list.append(nucleotide)

        wuss_list = change_letters(wuss_list, symbol_column_number + 1)
        bulge_loops = get_bulge_and_internal_loop_list(wuss_list)[1]
        bulge_indexes = []

        for bulge_loop in bulge_loops:
            interval = bulge_loop["intervals"][0]
            interval_range = list(range(interval[0], interval[1] + 1))
            bulge_indexes += interval_range

        for i in range(len(wuss_list)):
            nucleotide = wuss_list[i]
            if nucleotide[structure_column_number] == "indefinite_loop":
                if i + 1 in bulge_indexes:
                    wuss_list[i][structure_column_number] = "bulge_loop"
                else:
                    wuss_list[i][structure_column_number] = "internal_loop"

        return wuss_list
```

Рисунок 8. Функция load_ctwuss пакета RNAsselem.

Были разработаны функции, которые возвращают информацию о конкретных типах вторичной структуры в виде массивов, содержащих информацию о длине каждого из элементов вторичной структуры и интервалы с индексами начала и конца каждого из элементов. Такие функции были разработаны для всех типов вторичных структур, включая hairpin loop, internal loop, bulge loop, multifurcation loop, stem, external loop, функции соответственно называются *get_hairpin_loop_list*, *get_internal_loop_list*, *get_bulge_loop_list*, *get_multifurcation_loop_list*, *get_stem_list*, *get_external_loop_list*.
Каждая из этих функций возвращает список словарей, где i-й словарь содержит информацию об одном структурном элементе, такую как: длина элемента, массив с индексами начала и конца всех интервалов, содержащих данный элемент, однако у функции get_hairpin_loop_list есть дополнительный параметр – длина стебля, содержащего данную шпильку. В качестве входных данных эти функции принимают CT-WUSS файл или список, полученный с помощью функции load_ctwuss.

- Функция *get_hairpin_loop_list* (рис. 9) устроена следующим образом:

1) Алгоритм линейно проходит по каждому нуклеотиду в цепи РНК.
2) Находится непрерывная последовательность нуклеотидов с символом “_”.
3) С помощью функции *get_stem_list* находится длина стебля, содержащего данную шпильку.
4) Вся информация записывается в словарь, словарь добавляется в список.

```
694  def get_hairpin_loop_list(ctwuss):
695      """ Returns a list of hairpin loops with a dictionary of their properties.
696
697          Parameters:
698              ctwuss (str/list): path to the CT-modified file or loaded file
699
700          Returns:
701              hairpin_loops (list[dict]): list of the hairpin loop properties, including:
702                  - length - element length,
703                  - intervals - list of the start/stop positions,
704                  - sequences - DNA sequences at the intervals,
705                  - stem_length - length of the hairpin's stem.
706      """
707
708      wuss_list = load_ctwuss(ctwuss)
709      stems = get_stem_list(wuss_list)
710      hairpin_loops = []
711      length = 0
712
713      for i in range(len(wuss_list)):
714          nucleotide = wuss_list[i][structure_column_number]
715          if nucleotide == "hairpin_loop":
716              if length == 0:
717                  l_index = i + 1
718              length += 1
719          else:
720              if length != 0:
721                  r_index = i
722                  stem_dictionary = get_interval(stems, l_index - 1)
723                  stem_length = stem_dictionary["length"]
724                  d = get_structure_dict(wuss_list, length, [[l_index, r_index]])
725                  d["stem_length"] = stem_length
726                  hairpin_loops.append(d)
727                  length = 0
728
729      return hairpin_loops
```

Рисунок 9. Функция get_hairpin_loop_list пакета RNAsselem.

- Функция *get_internal_loop_list* использует внутреннюю функцию *get_bulge_and_internal_loop_list*, описанную раннее.

- Функция *get_bulge_loop_list* использует внутреннюю функцию *get_bulge_and_internal_loop_list*, описанную раннее.

- Функция *get_multifurcation_loop_list* устроена следующим образом:
  1) Из списка, содержащего всю цепь РНК, полученного с помощью функции *load_ctwuss*, поочередно, в зависимости от уровня вложенности, удаляются нуклеотиды, принадлежащие стеблю и нуклеотиды, находящиеся внутри стебля, такие как шпильки, выпячивания и внутренние петли. Первыми удаляются все нуклеотиды стеблей первого (внешнего) уровня вложенности, обозначающиеся символом “<” или “>”, вторыми – обозначающиеся символом “(“ или “)”, третьими - обозначающиеся символом “[“ или “]”, четвертыми - обозначающиеся символом “{“ или “}”.
  2) После удаления каждого из уровней, последовательность подряд идущих нуклеотидов с символом “,” являются ровно одной многоразветленной петлей. Информация о данном элементе записывается в словарь и добавляется в список многоразветвленных петель, после чего все нуклеотиды, принадлежащие данному элементу также удаляются.
  3) Алгоритм продолжает работу, пока не будут удалены все уровни.

```
857  def get_multifurcation_loop_list(ctwuss):
858      """ Returns a list of multifurcation loops with a dictionary of their properties.
859
860          Parameters:
861              ctwuss (str/list): path to the CT-modified file or loaded file
862
863          Returns:
864              multifurcation_loops (list[dict]): list of the multifurcation loop properties, including:
865               - length - element length,
866               - intervals - list of the start/stop positions,
867               - sequences - DNA sequences at the intervals.
868
869      """
870
871      wuss_list = load_ctwuss(ctwuss)
872      wuss_list_copy = deepcopy(wuss_list)
873
          Marat Kazanov
874      def find_bracket(bracket, array):
875          for i in range(len(array)):
876              line = array[i]
877              if bracket in line:
878                  return i
879                  break
880          else:
881              return -1
```

Рисунок 10. Функция get_multifurcation_loop_list. строки пакета RNAsselem 857 – 881.

```
FedorKazanov +1
def delete_bracket(array, bracket_index):

    beginning = array[bracket_index]
    connection_number = beginning[connection_column_number] - 1
    r = array[bracket_index:]
    start_index = 1
    previous_index = -1

    m_nucleotides = []

    for i in range(len(r)):
        nucleotide = r[i]
        if nucleotide[structure_column_number] == "multifurcation_loop":
            m_nucleotides.append(nucleotide)

        if nucleotide[index_column_number] - 1 == connection_number:
            last_index = bracket_index + i
            break

    if len(m_nucleotides) == 1:
        intervals = [[m_nucleotides[0][0], m_nucleotides[0][0]]]
    else:
        intervals = []
        for i in range(len(m_nucleotides)):

            nucleotide = m_nucleotides[i]
            index = nucleotide[index_column_number] - 1

            if i == 0:
                start_index = index
            else:
                if index - previous_index != 1:
                    intervals.append([start_index + 1, previous_index + 1])
                    start_index = index
                if i == len(m_nucleotides) - 1:
                    intervals.append([start_index + 1, index + 1])

            previous_index = index

    del array[bracket_index: last_index + 1]
    multifurcation_loop = get_structure_dict(wuss_list, len(m_nucleotides), intervals)

    return [array, multifurcation_loop]

multifurcation_loops = []
for bracket in ["<", "(", "[", "{"]:
    while True:
        bracket_index = find_bracket(bracket, wuss_list_copy)
        if bracket_index < 0:
            break
        wuss_list_copy, multifurcation_loop = delete_bracket(wuss_list_copy, bracket_index)
        if multifurcation_loop["length"] != 0:
            multifurcation_loops.append(multifurcation_loop)
multifurcation_loops.sort(key=lambda x: x["intervals"][0][0])

return multifurcation_loops
```

Рисунок 11. Функция get_multifurcation_loop_list. строки пакета RNAsselem 883 – 938.

- Функция *get_stem_loop_list* устроена следующим образом:
  1) Алгоритм линейно проходит по каждому нуклеотиду в цепи РНК.
  2) Находятся подряд идущие нуклеотиды, принадлежащие стеблю, с одинаковым символом открывающейся скобки, при этом, если они прерываются выпячиваниями или внутренними петлями, они все равно считаются идущими подряд.
  3) Найденному участку находится соответствующий комплементарный участок. Два данных участка составляют полный элемент стебля, информация о котором записывается в словарь и добавляется в список.

```
635     # public
        FedorKazanov +1
636     def get_stem_list(ctwuss):
637         """ Returns a list of stems with a dictionary of their properties.
638
639             Parameters:
640                 ctwuss (str): path to the CT-modified file or loaded file
641
642             Returns:
643                 stem_list (list[dict]): list of the with stem properties, including:
644                  - length - element length,
645                  - intervals - list of the start/stop positions,
646                  - sequences - DNA sequences at the intervals.
647
648         """
649
650         wuss_list = load_ctwuss(ctwuss)
651         pairs = [["(", ")"], ["[", "]"], ["<", ">"], ["{", "}"]]
652         stem_list = []
653         length = 0
654         left_index = -1
655
656         intervals = []
657         for pair in pairs:
658             bracket = pair[0]
659             cl_bracket = pair[1]
660             for i in range(len(wuss_list)):
661                 nucleotide = wuss_list[i]
662
663                 if nucleotide[symbol_column_number] == bracket:
664                     previous_nucleotide = wuss_list[i - 1]
665                     next_nucleotide = wuss_list[i + 1] if i != len(wuss_list) - 1 else 8 * [-3]
666                     if length == 0:
667                         end_index = nucleotide[connection_column_number] - 1
668                     length += 1
669                     if previous_nucleotide[symbol_column_number] != bracket:
670                         left_index = nucleotide[index_column_number] - 1
671                     if next_nucleotide[symbol_column_number] != bracket:
672                         intervals.append([left_index + 1, nucleotide[index_column_number]])
673
674                 if nucleotide[symbol_column_number] == cl_bracket:
675                     previous_nucleotide = wuss_list[i - 1]
676                     next_nucleotide = wuss_list[i + 1] if i != len(wuss_list) - 1 else 8 * [-3]
677                     length += 1
678                     if previous_nucleotide[symbol_column_number] != cl_bracket:
679                         left_index = nucleotide[index_column_number] - 1
680                     if next_nucleotide[symbol_column_number] != cl_bracket:
681                         intervals.append([left_index + 1, nucleotide[index_column_number]])
682                     if nucleotide[index_column_number] - 1 == end_index:
683                         d = get_structure_dict(wuss_list, length, intervals)
684                         stem_list.append(d)
685                         length = 0
686                         intervals = []
687
688         stem_list.sort(key=lambda x: x["intervals"][0][0])
689
690         return stem_list
```

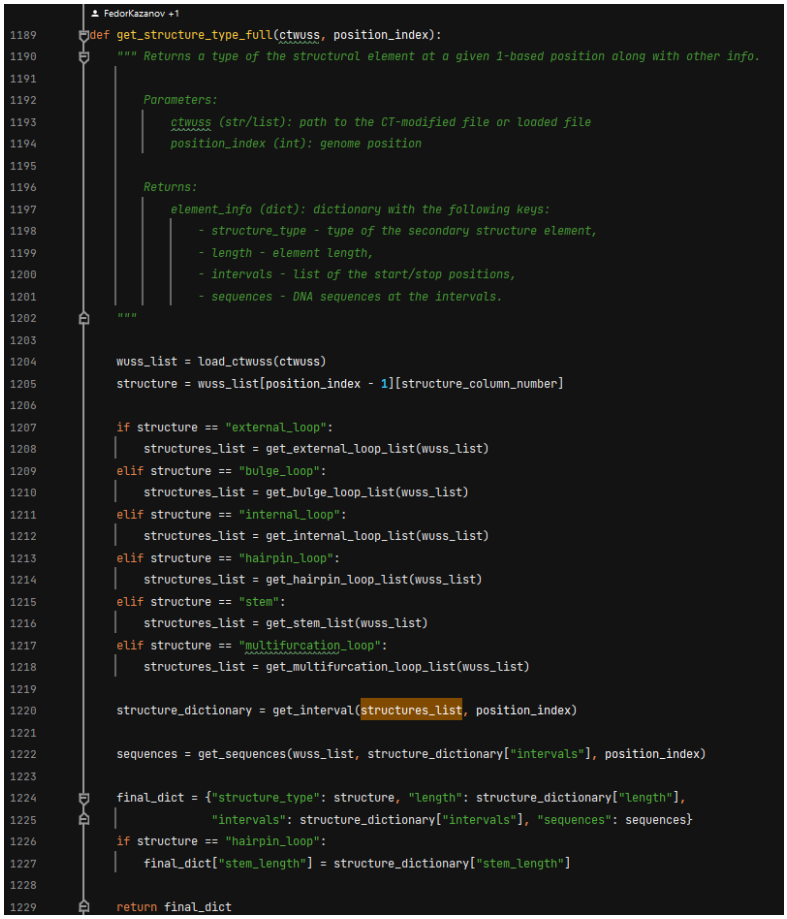
Рисунок 12. Функция get_stem_list пакета RNAsselem.

- Функция *get_external_loop_list* устроена следующим образом:
  1) Алгоритм линейно проходит по каждому нуклеотиду в цепи РНК.
  2) Находится непрерывная последовательность нуклеотидов с символом “:”, информация о данном элементе записывается в словарь, словарь добавляется в список.

```
555  def get_external_loop_list(ctwuss):
556      """ Returns a list of external loops with a dictionary of their properties.
557
558          Parameters:
559          ctwuss (str/list): path to the CT-modified file or loaded file
560
561          Returns:
562              external_loops (list[dict]): list of the external loop properties, including:
563                  - length - element length,
564                  - intervals - list of the start/stop positions,
565                  - sequences - DNA sequences at the intervals.
566      """
567
568      wuss_list = load_ctwuss(ctwuss)
569      external_loops = []
570      length = 0
571
572      for i in range(len(wuss_list)):
573          nucleotide = wuss_list[i][structure_column_number]
574
575          if nucleotide == "external_loop":
576              if length == 0:
577                  l_index = i + 1
578              length += 1
579          else:
580              if length != 0:
581                  r_index = i
582                  d = get_structure_dict(wuss_list, length, [[l_index, r_index]])
583                  external_loops.append(d)
584                  length = 0
585
586      if length != 0:
587          r_index = len(wuss_list)
588          d = get_structure_dict(wuss_list, length, [[l_index, r_index]])
589          external_loops.append(d)
590
591      return external_loops
```

Рисунок 13. Функция get_external_loop_list пакета RNAsselem.

- Была разработана функция *get_structure_type_full*, которая по номеру позиции нуклеотида возвращает данные об элементе вторичной структуры, включающим в себя данный нуклеотид: тип элемента вторичной структуры, длина элемента вторичной структуры, интервалы с индексами начала и конца данного элемента вторичной структуры, строки с фрагментами цепи РНК этого элемента вторичной структуры.

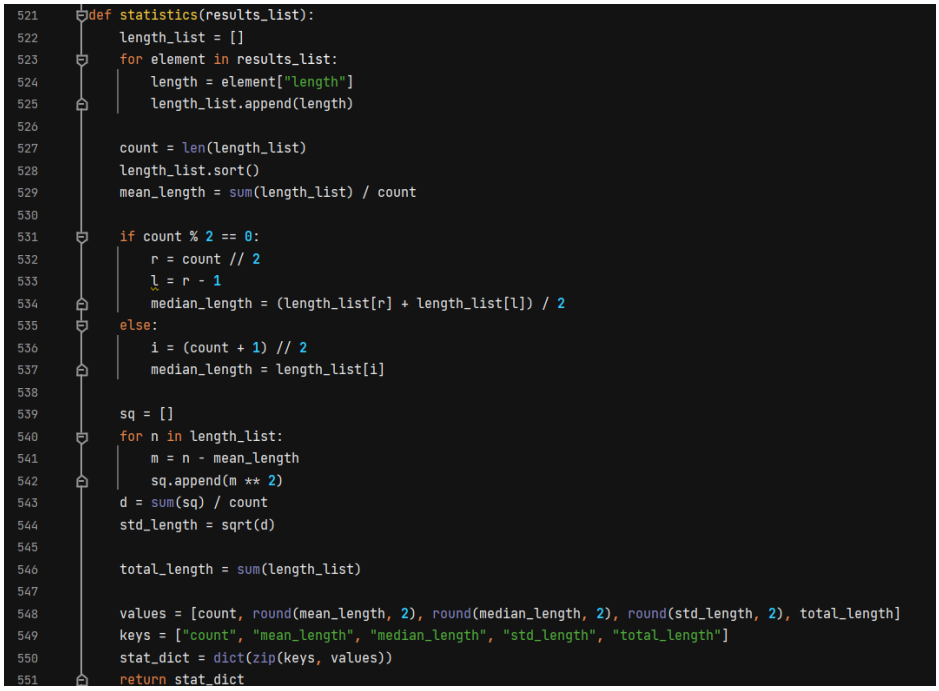


```
FedorKazanov +1
def get_structure_type_full(ctwuss, position_index):
    """ Returns a type of the structural element at a given 1-based position along with other info.

        Parameters:
            ctwuss (str/list): path to the CT-modified file or loaded file
            position_index (int): genome position

        Returns:
            element_info (dict): dictionary with the following keys:
                - structure_type - type of the secondary structure element,
                - length - element length,
                - intervals - list of the start/stop positions,
                - sequences - DNA sequences at the intervals.
    """

    wuss_list = load_ctwuss(ctwuss)
    structure = wuss_list[position_index - 1][structure_column_number]

    if structure == "external_loop":
        structures_list = get_external_loop_list(wuss_list)
    elif structure == "bulge_loop":
        structures_list = get_bulge_loop_list(wuss_list)
    elif structure == "internal_loop":
        structures_list = get_internal_loop_list(wuss_list)
    elif structure == "hairpin_loop":
        structures_list = get_hairpin_loop_list(wuss_list)
    elif structure == "stem":
        structures_list = get_stem_list(wuss_list)
    elif structure == "multifurcation_loop":
        structures_list = get_multifurcation_loop_list(wuss_list)

    structure_dictionary = get_interval(structures_list, position_index)

    sequences = get_sequences(wuss_list, structure_dictionary["intervals"], position_index)

    final_dict = {"structure_type": structure, "length": structure_dictionary["length"],
                  "intervals": structure_dictionary["intervals"], "sequences": sequences}
    if structure == "hairpin_loop":
        final_dict["stem_length"] = structure_dictionary["stem_length"]

    return final_dict
```

Рисунок 14. Функция get_structure_type_full пакета RNAsselem.

Описательная статистика элементов вторичной структуры

- Была разработана внутренняя функция *statistics*, которая для конкретного типа элементов вторичной структуры возвращает различные статистические данные, такие как количество элементов конкретного типа вторичной структуры, средняя длина, медиана длины, стандартное отклонение длины элемента конкретного типа вторичной структуры и общая длина всех элементов конкретного типа вторичной структуры.



```python
def statistics(results_list):
    length_list = []
    for element in results_list:
        length = element["length"]
        length_list.append(length)

    count = len(length_list)
    length_list.sort()
    mean_length = sum(length_list) / count

    if count % 2 == 0:
        r = count // 2
        l = r - 1
        median_length = (length_list[r] + length_list[l]) / 2
    else:
        i = (count + 1) // 2
        median_length = length_list[i]

    sq = []
    for n in length_list:
        m = n - mean_length
        sq.append(m ** 2)
    d = sum(sq) / count
    std_length = sqrt(d)

    total_length = sum(length_list)

    values = [count, round(mean_length, 2), round(median_length, 2), round(std_length, 2), total_length]
    keys = ["count", "mean_length", "median_length", "std_length", "total_length"]
    stat_dict = dict(zip(keys, values))
    return stat_dict
```

Рисунок 15. Функция statistics пакета RNAsselem.

- Для удобства применения программного пакета были добавлены отдельные функции для вычисления описательной статистики для каждого из структурных элементов определённого типа. Функции называются *get_stem_stat*, *get_external_loop_stat*, *get_internal_loop_stat*, *get_bulge_loop_stat*, *get_multifurcation_loop_stat*, *get_haitpin_loop_stat*. В качестве входных данных каждая из функций принимает CT-WUSS файл или массив с информацией из CT-WUSS файла. Каждая из этих функций сначала вызывает функцию для получения списка с конкретными элементами данной вторичной структуры, затем передает полученный список в функцию statistics, после чего возвращает список со статистическими данными.

- Была разработана функция save_stat для сохранения всех статистических данных для всех типов элементов вторичной структуры РНК вируса в качестве текстового файла.

```python
def save_stat(ctwuss_path, output_path, is_append=False):
    """ Saves statistics for all types of structural elements to file.

    Calculates and saves statistics on hairpin loops, internal loops, bulges, multifurcation loops and external loops to a specified output file.

    Parameters:
        ctwuss_path (str): path to the CT-modified file or loaded file
        output_path (str): path to the output file
        is_append (bool): append output to existing file or create a new file
    """

    lines = ["External_loops", "Hairpin_loops", "Internal_loops", "Bulge_loops", "Multifurcation_loops", "Stems"]
    stat = [get_external_loop_stat(ctwuss_path), get_hairpin_loop_stat(ctwuss_path),
            get_internal_loop_stat(ctwuss_path),
            get_bulge_loop_stat(ctwuss_path), get_multifurcation_loop_stat(ctwuss_path), get_stem_stat(ctwuss_path)]

    if isfile(output_path) and not is_append:
        print("This file is already exist. Delete this file and try again.")
        return -1

    output_file = open(output_path, "a")
    if is_append:
        output_file.write("\n" * 2)

    line1 = basename(ctwuss_path)
    output_file.write(line1 + "\n")
    line2 = "\tCount\tMean_length\tMedian_length\tStd_length\tTotal_length\n"
    output_file.write(line2)

    for i in range(6):
        line = lines[i] + "\t"
        stat_values = list(stat[i].values())
        for i in range(5):
            stat_values[i] = str(stat_values[i])
        line += "\t".join(stat_values) + '\n'
        output_file.write(line)
    return 0
```

Рисунок 16. Функция save_stat пакета RNAsselem.

- Были написаны функции *get_paired* и *get_unpaired* для подсчета количества спаренных и неспаренных нуклеотидов в РНК.

```python
def get_unpaired(ctwuss):
    """ Returns the number of unpaired nucleotides in genome.

        Parameters:
            ctwuss (str/list): path to the CT-modified file or loaded file

        Returns:
            unpaired_cnt (int): number of unpaired nucleotides
    """

    wuss_list = load_ctwuss(ctwuss)
    connection_indexes = get_columns(wuss_list, connection_column_number)
    unpaired_cnt = connection_indexes.count(0)
    return unpaired_cnt
```

Рисунок 17. Функция get_unpaired пакета RNAsselem.

```python
def get_paired(ctwuss):
    """ Returns the number of paired nucleotides in genome.

        Parameters:
            ctwuss (str/list): path to the CT-modified file or loaded file

        Returns:
            paired_cnt (int): number of paired nucleotides
    """

    total_length = get_total_length(ctwuss)
    unpaired = get_unpaired(ctwuss)
    paired_cnt = total_length - unpaired
    return paired_cnt
```

Рисунок 18. Функция get_paired пакета RNAsselem.

Результаты

Разработанный пакет для анализа вторичной структуры РНК RNAsselem был помещен в репозиторий сервера Github по адресу https://github.com/KazanovLab/RNAsselem для предоставления возможности общего доступа к пакету, ведения версий пакета и протоколирования ошибок, замечаний и предложений (Рис 19).

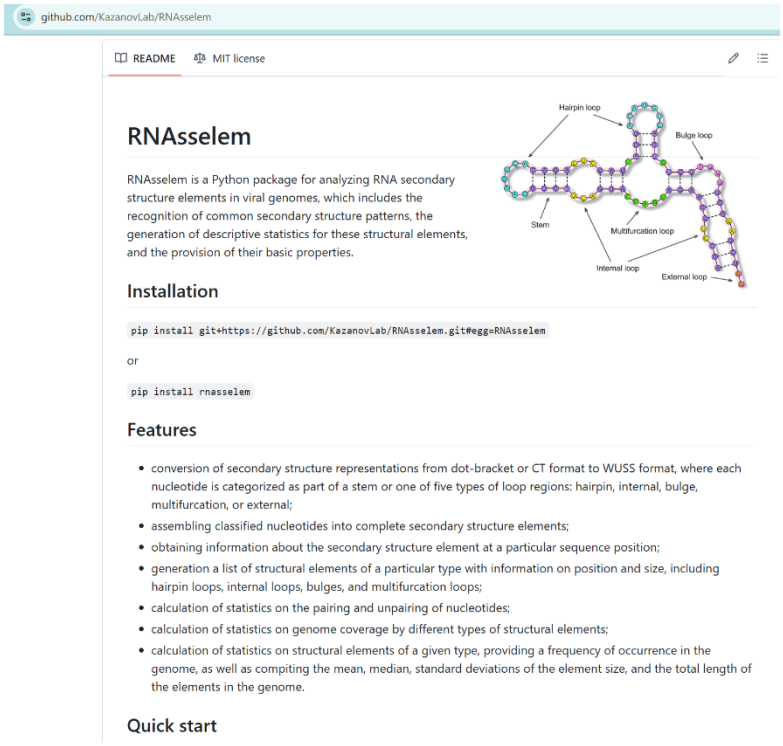


Рисунок 19. Пакет RNAaasselem на сервере Github.

Для возможности свободной установки для всех пользователей, пакет был размещен в репозитории Python Package index (PyPi), предназначенном для хранения и распространения пакетов на языке Python (Рис 20).

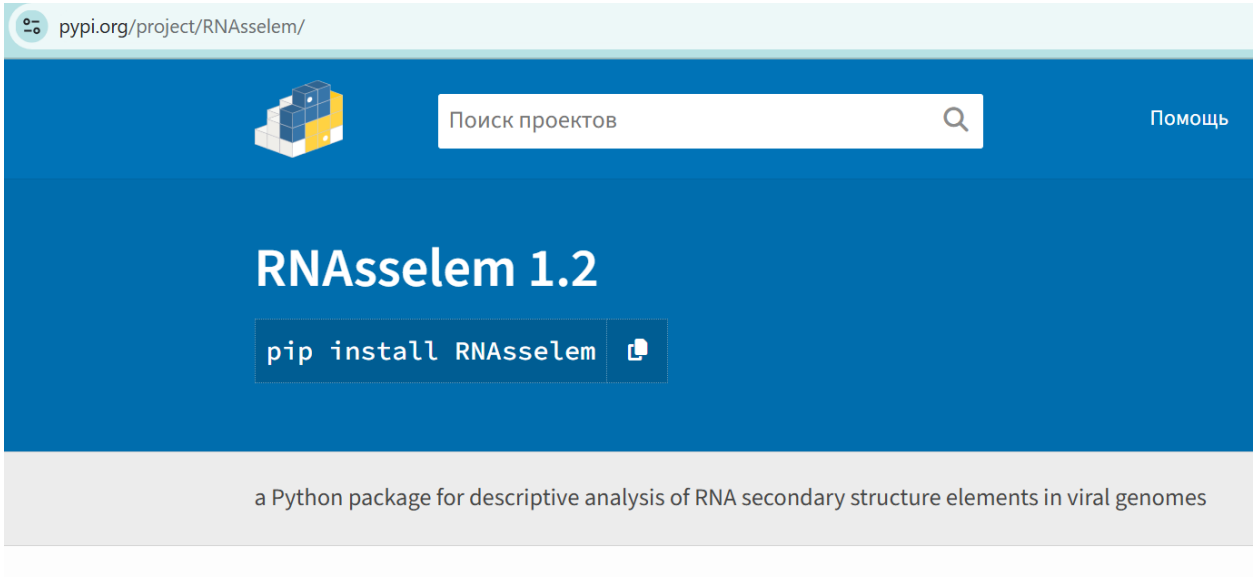


Рисунок 20. Пакет RNAsselem в репозитории PyPi.

Разработанный мной пакет был применен сотрудниками центра молекулярной и клеточной биологии Сколковского университета науки и технологий для анализа вируса Денге, трех подвидов вируса гепатита С и четырех штаммов коронавируса SARS-CoV-2. По результатам анализа сотрудниками была опубликована статья в международном рецензированном научном журнале Scientific Reports (Рис 21).

Рисунок 21. Страница публикации научного проекта с применением пакета RNAsselem в научном журнале Scientific reports.