Домашнее задание 3

Для выполнения задания я разработал последовательность действий, которая включает загрузку данных, их обработку, анализ и автоматизацию с помощью CWL. Все шаги были выполнены на Fedora Linux.

1. **Загрузка данных секвенирования**
   Скачал данные секвенирования из базы NCBI под идентификатором SRR31569776. Для этого использовал утилиту prefetch из пакета sra-toolkit, который предварительно установил через менеджер пакетов Fedora DNF:

```
prefetch SRR31569776
```

2. **Подготовка референсного генома**
   Далее я скачал референсный геном hg38.fa вместе с необходимыми индексными файлами (.amb, .ann, .bwt, .pac, .sa), которые требуются для выравнивания.
3. **Установка инструментов**
   Для выполнения анализа я установил инструменты через DNF.
   - FastQC — для проверки качества прочтений.
   - bwa — для выравнивания последовательностей.
   - samtools — для работы с файлами выравнивания.

   Команда:

```
sudo dnf install fastqc bwa samtools
```

4. **Обработка данных секвенирования**
   Используя fastq-dump из sra-toolkit, я разделил данные из файла SRR31569776 на парные прочтения:

```
fastq-dump --split-files SRR31569776
```

5. **Выравнивание последовательностей**

   Для выравнивания прочтений на референсный геном я применил bwa mem:

```
bwa mem -t 4 hg38.fa SRR31569776_1.fastq
SRR31569776_2.fastq > alignment.sam
```

Затем с помощью samtools отсортировал полученный файл и преобразовал его в бинарный формат:

```
samtools sort alignment.sam -o alignment_sorted.bam
```

6. **Индексация выравнивания**

   Чтобы оптимизировать дальнейшую работу с файлом alignment_sorted.bam, я создал индекс:

```
samtools index alignment_sorted.bam
```

7. **Анализ качества выравнивания**

   Я написал скрипт на Python, который автоматизирует анализ результатов выравнивания. Скрипт выполняет команду samtools flagstat, извлекает процент картированных ридов и сравнивает его с порогом 95%. Результат составил 99.89%, что подтверждает высокое качество выравнивания.

8. **Инструкция по установке CWL на Fedora Linux**
   Для создания пайплайна я установил фреймворк CWL с помощью Python:

```
python3 -m pip install --user pipx
pipx install cwltool
```

Добавил путь ~/.local/bin в PATH, выполнив

```
export PATH=$PATH:~/.local/bin
```

И, для постоянного эффекта, сохранил это в

```
source ~/.bashrc
```

9. **Тестирование CWL**
   Для проверки работы CWL я создал тестовый файл hello_world.cwl, который выводит сообщение "Hello, CWL!" при запуске:

```
cwltool hello_world.cwl --message "Hello, CWL!"
```

10. **Создание пайплайна CWL**
    Я разработал три файла CWL:
    - bwa-align.cwl — для выравнивания с помощью bwa mem.
    - samtools-flagstat.cwl — для анализа с использованием samtools flagstat.
    - workflow.cwl — объединяющий шаги в единый рабочий процесс.

    Пайплайн был протестирован с помощью cwltool и успешно выполнил все задачи.
11. **Визуализация рабочего процесса**

    Для создания графического представления я использовал команду:

```
cwltool --print-dot workflow.cwl | dot -Tpng -o
workflow.png
```

Полученный файл workflow.png демонстрирует структуру пайплайна в виде направленного ациклического графа (DAG), отображая этапы align и flagstat.

12. **Описание визуализации**
    DAG иллюстрирует поток данных от входных файлов (ref_genome, read1, read2) через выравнивание и анализ к выходному файлу flagstat_output. В отличие от традиционной блок-схемы, DAG не содержит циклов, подчеркивая линейную последовательность шагов.
13. **Вывод**
    Задание выполнено полностью с использованием Fedora Linux. Я создал функциональный пайплайн в формате CWL, скрипт для анализа качества выравнивания и визуализацию процесса. Крупные файлы, не приложены из-за их размера. Высокий процент картированных ридов (99.89%) подтверждает качество выполненной работы и превышает установленный порог в 95%.