Практикум 14

1.Скачивание чтений

wget
ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR424/009/SRR4240379/SRR4240379.fastq.gz

Последние три символа 379 преобразовываются в 009, так как EBI FTP для FASTQ файлов берет сначала первые 5 символов, затем - последние три и преобразовывает их на сервере
Из FastQC: 7 400 155 чтений, длина - 36 нуклеотидов
2. Подготовка чтений программой trimmomatic.
Создаем файл, который содержит последовательность адаптеров illumina
cat /mnt/scratch/NGS/adapters/*.fa > adapters_all.fasta
Теперь используем trimmonatic:

```
TrimmomaticSE \
  -phred33 \
  SRR4240379.fastq.gz \
  SRR4240379_trimmed.fastq.gz \
  ILLUMINACLIP:adapters_all.fasta:2:7:7 \
  TRAILING:20 \
  MINLEN:32
```

Где параметры означают:

adapters_all.fasta - файл, содержащий последовательности адаптеров Illumina (мы собрали его командой cat /mnt/scratch/NGS/adapters/*.fa > adapters_all.fasta).

2 - минимальный размер seed (начальный фрагмент для поиска совпадения).

7 - порог совпадения (palindrome clip threshold): если прочитанный фрагмент и адаптер совпадают на ≥7 нуклеотидов, чтение обрезается.

7 - простой порог совпадения (simple clip threshold): то же самое, но для одностороннего совпадения.

TRAILING:20 - команда просматривает чтение с конца и удаляет нуклеотиды, пока не встретит позицию с качеством ≥ Q20.

MINLEN:32 - удаляет все чтения, ставшие короче 32 нуклеотидов.

Входные данные: SRR4240379.fastq.gz

Вывод: SRR4240379_trimmed.fastq.gz

Теперь прогоняем файлы SRR4240379.fastq.gz
SRR4240379_trimmed.fastq.gz через fastqc, и в html файлах исходное число чтений вышло 7400155, после обработки - 6974267, то есть осталось около 94,24% последовательностей чтений, удалено - 425888 (5.76%)

3. Сборка генома (Velvet)

Для создания сборки de novo мы применяем две программы пакета Velvet, которые, используя граф Де Брёйна, строят сборку из коротких чтений. Сперва применяем программу velveth, которая создаёт список k-меров длиной 31 нуклеотид, встретившихся в наших чтениях:

velveth velvet_output 31 -short -fastq SRR4240379_trimmed.fastq.gz

Где:

- velvet_output – новая директория, куда программа положит полученные файлы;
- 31 – длина k-меров;
- -short – короткие и непарные чтения;
- -fastq – формат входного файла.

Следующий этап - создание самой сборки на основе графа Де Брёйна, для этого используем программу velvetg:

etg velvet_output

Результаты по нашей сборке:

- N50 = 25 646 нуклеотидов (по скрипту для расчёта N50).
- Длины трёх самых длинных контигов: 49 912, 49 262 и 33 085 нт.
- Покрытие трёх самых длинных контигов: 35.907238, 34.772179 и 36.259029.
- Длина трёх самых коротких контигов: 31 нт.
- Покрытие трёх самых коротких контигов: от 2.7 до 3.1 (например, 2.709677, 3.000000, 3.096774).
- Медианное покрытие по контигам: около 35 чтений.

Аномальное покрытие:

- Есть несколько контигов с покрытием значительно выше медианного (около 172.5, 177.2, 181.7 и максимум 458.4 при длине 282 нт, то есть более чем в 5 раз выше медианы).
- Есть контиги с покрытием около 2–3, то есть более чем в 5 раз ниже медианного покрытия.

Для анализа длин и покрытий использовалась команда:

bash

grep "^>" contigs.fa | tr "_" "\t" | cut -f4,5,6 | sort -n | less

Она выводит длину и покрытие каждого контига в виде <длина>\tcov\t<покрытие, отсортированные по длине.
Покрытие для самых длинных контигов находится вблизи медианного значения, поэтому их можно считать надёжными и использовать для дальнейшего анализа.

Скрипт для н50:

```
#!/bin/bash


lengths=""

while read line; do

  if [[ $line == ">"* ]]; then

    len=${line#*length_}


    len=${len%%_*}

    lengths="$lengths$len "
  fi
done < contigs.fa


sorted=($(for num in $lengths; do echo $num; done | sort -nr))

total=0
for num in "${sorted[@]}"; do
  total=$((total + num))
done


half=$((total / 2))
sum=0
n50=0

for num in "${sorted[@]}"; do
  sum=$((sum + num))
  if [ $sum -ge $half ]; then
    n50=$num
    break
```

```
  fi
done

echo "Всего контигов: ${#sorted[@]}"
echo "Общая длина сборки: $total нуклеотидов"
echo "Половина от общей длины: $half нуклеотидов"
echo "N50 = $n50 нуклеотидов"
echo ""
echo "L50 = $(echo "$sum >= $half" | bc) контигов (последний был номер $n50)"
```

Вывод: N50 = 25646 нуклеотидов

len=${line#*length_}

- Удаляет всё от начала строки ДО первого вхождения length_
- Пример: ">NODE_1_length_33062_cov_32.52" → "33062_cov_32.52"

2. len=${len%%_*}

- Удаляет всё с первого символа _ и до конца строки
- Пример: "33062_cov_32.52" в "33062"

3. sorted=($(...))

- $(...) выполняет команду внутри скобок
- sort -nr сортирует числа (-n) в обратном порядке (-r)
- Внешние скобки (...) создают массив из результата

4. ${#sorted[@]}

- Возвращает количество элементов в массиве sorted

4. полный геном штамма *Buchnera aphidicola*
GenBank: NC_002528.1
Бластуем с первым контигом, выделив его последовательность и вставив в бласт вместе с AC штамма

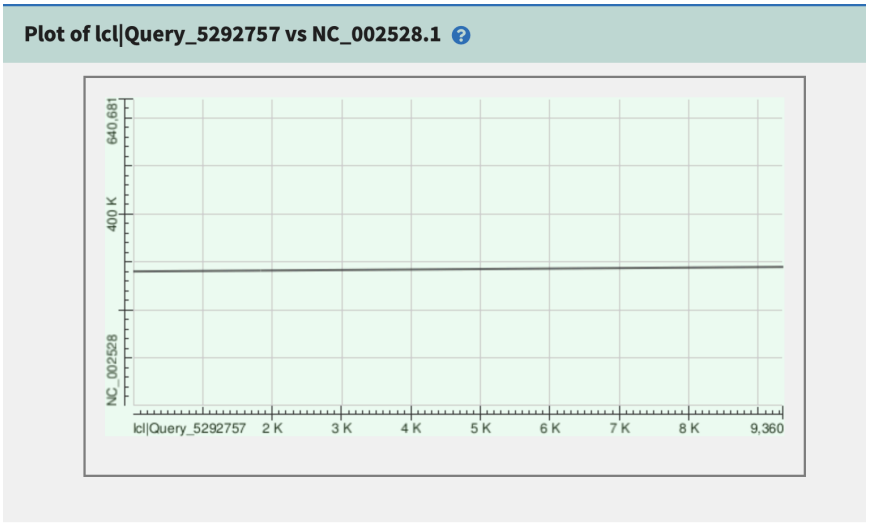

Рис.!. Выравнивание с первым контигом

То есть, контиг полностью идентичен участку референса, не содержит ошибок сборки (разрывов, дубликатов), корректно "ложится" на геном

Позиция на эталонной хромосоме NC_002528.1 -примерно от 290000 до 300000 bp

Позиция на контиге (Query) - От 0 до 9360 bp

Со вторым:

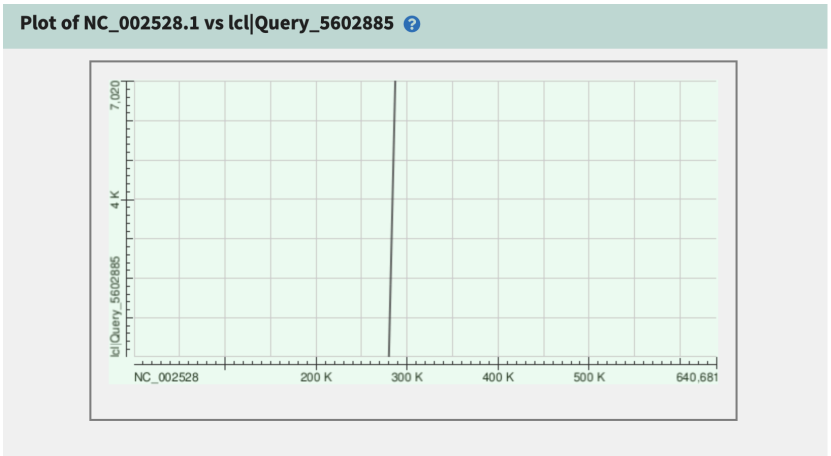

Рис.2. Выравнивание со вторым контигом
Результат идентичен предыдущему, все совпадает, контигложится на геном как единый непрерывный фрагмент.

Позиция на эталонной хромосоме NC_002528.1 - От 280000 до 290000 bp

Позиция на контиге (Query) - От 0 до 7,020 bp

Выравниваем с третим:

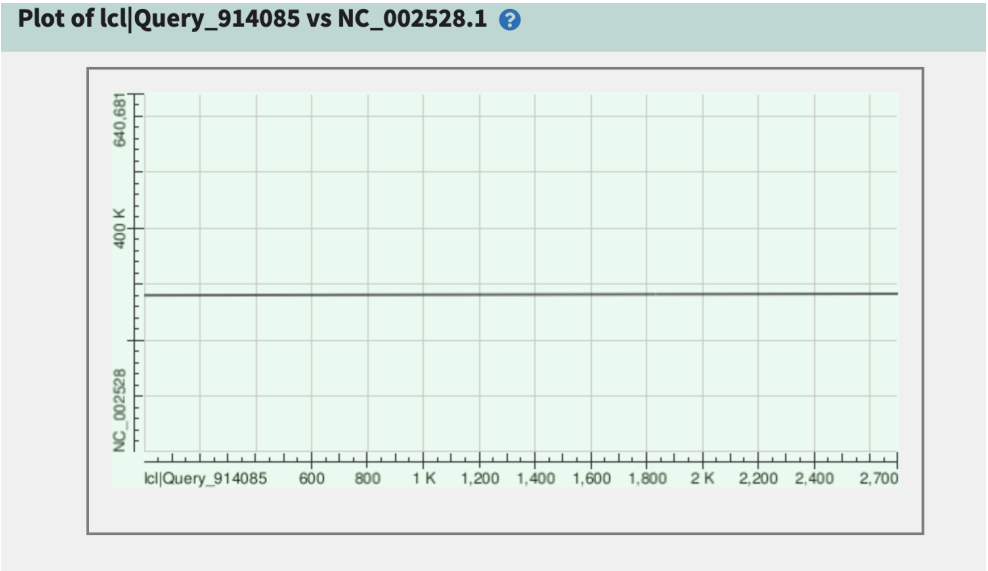

Рис.3. Выравнивание с третьим контигом
Результат тот же

Позиция на эталонной хромосоме NC_002528.1 - От 280000 до 290000 bp

Позиция на контиге (Query) - От 0 до 2700 bp