Практикум №14

1. Подготовка чтений программой trimmomatic

Так как мы работали с одиночными чтениями, то среди адаптеров в директории **/mnt/scratch/NGS/adapters** были отобраны те, что содержали аббревиатуру SE — Single-End (одноконцевые чтения):

TruSeq2-SE.fa
TruSeq3-SE.fa

Они были объединены в один файл, после чего было проведено триммирование:

```
cat /mnt/scratch/NGS/adapters/TruSeq2-SE.fa
/mnt/scratch/NGS/adapters/TruSeq3-SE.fa > adapters.fa
```

```
TrimmomaticSE -phred33 -threads 4 SRR1722713.fastq.gz
SRR1722713 _trimmed_adapters.fastq.gz
ILLUMINACLIP:adapters.fa:2:7:7
```

Всего чтений 13241431 обработано. Было удалено адаптеров 2737 **(0.02%)**. Размер файла до очистки: **0.86 Гб**, после: **0.84 Гб**.

После исключения адаптеров, из чтений были удалены нуклеотиды с качеством ниже 20 с 3' конца и отфильтрованы чтения длины меньше 32 нуклеотидов:

```
TrimmomaticSE -phred33 -threads 4
SRR1722713_trimmed_adapters.fastq.gz
SRR1722713_trimmed_adapters_quality.fastq.gz MINLEN:32
TRAILING:20
```

В результате выполнения команды было отброшено 216327 (1.63%) чтений.

2. Подготовка k-меров на основе чтений

Для построения 31-меров для одноконцевых чтений длиной 101 нуклеотид была использована следующая команда:

```
velveth velveth_out 31 -fastq.gz -short
SRR1722713_trimmed_adapters_quality.fastq.gz
```

В результате работы программы внутри рабочей директории были созданы файлы Log (текстовый файл с параметрами запуска команд), Roadmaps (показывают расположение k-меров в чтениях) и Sequences (база данных всех чтений).

3. Сборка на основе генома

Следующим шагом были получены контиги генома хлоропласта с помощью программы velvetg, применённой к ранее полученной директории без дополнительных параметров:

```
velvetg velveth_out
```

Используя Python были получены значения L50 и N50 (см. Приложение).

Из выдачи:

N50 = 142 нуклеотида
L50 = 15586 контига

Такие плохие качества сборки объясняются скорее всего тем, что в чтениях имелись как РНК хлоропласта, так и РНК ядра, это также становится очевидным, если посмотреть на три самый длинных контига и на их покрытия:

```
grep "^>" contigs.fa | awk -F"[ _]" '{print $4, $6, $0}' | sort -nr | head -3 | awk '{print "Conting:", $4, "\nLength:", $1, "bp\nCover:", $2, "x\n"}'
```

Выдача:

```
Conting:
Length: 1899 bp
Cover: 7.360716 x

Conting:
Length: 1664 bp
Cover: 5.924279 x

Conting:
Length: 1609 bp
Cover: 8.435053 x
```

Для получения сборки хлоропластного генома был сделан фильтр по покрытию, чтобы отобрать только те контиги, которые имеют высокий показатель покрытия.

Для этого программе velveth были поданы дополнительные аргументы:

```
velvetg velveth_new_out -exp_cov auto -cov_cutoff 70
```

-exp_cov auto позволяет программе определять ожидаемое покрытие для сборки, что может помочь в разрешении сложных ситуаций

-cov_cutoff 70 удаляет контиги с покрытием меньше 70

Для оценки качества чтений был применён крипт на Python:

N50 = 816 нуклеотида
L50 = 45 контига

Из выдачи следует, что качество заметно улучшилось.

Три самых длинных контига:

```
Conting:
Length: 2738 bp
Cover: 1534.176392 x

Conting:
Length: 2383 bp
Cover: 463.631989 x

Conting:
Length: 2120 bp
Cover: 1306.621216 x
```

Также были рассмотрены по 3 контига с наибольшими и наименьшими покрытиями:

```
Conting: >NODE_26_length_32_cov_228471.718750
Cover: 228471.718750 x
Length: 32 bp

Conting: >NODE_82_length_65_cov_173093.453125
Cover: 173093.453125 x
Length: 65 bp

Conting: >NODE_1_length_64_cov_71757.625000
Cover: 71757.625000 x
Length: 64 bp

Conting: >NODE_1033_length_85_cov_79.282356
Cover: 79.282356 x
Length: 85 bp

Conting: >NODE_1187_length_31_cov_80.516129
Cover: 80.516129 x
Length: 31 bp
```

```
Conting: >NODE_633_length_107_cov_80.785049
Cover: 80.785049 x
Length: 107 bp
```

При сравнении контига с наибольшим и наименьшим покрытием выясняется, что они различаются примерно в 2,9*10^3 раз. Скорее всего контиги с самым большим покрытием собраны из повторяющихся участков. Это может указывать на то, что они были ошибочно собраны как отдельные контиги.

Среди контигов с наименьшим покрытием данное значение находится на уровне не ниже 79, что говорит о хорошем покрытии.

4. Гомология с геномом Arabidopsis thaliana

Для начала полученные контиги были последовательно поданы в megablast для поиска по банку "RefSeq Genome Database", ограниченному на вид *Arabidopsis thaliana*.

Все три контига были сопоставлены на хлоропластной хромосоме данного вида с идентификатором NC_000932.1.

После чего проводилось парное выравнивание каждого контига с найденной хлоропластной хромосомой, результаты которых приведены ниже.

Рис. 1. Dot Plot Conting 2738

Характеристики:

Score	Expect	Identities	Gaps	Strand
4942 bits(2676)	0.0	2741/2769(99%)	18/2769(0%)	Plus/Minus

Из описания выравнивания ясно, что контиг длинной 2738 нуклеотидов является почти полностью идентичным фрагменту хлоропластной хромосомы.

Контиг был выровнен от 52235 до 54987 нуклеотида.

Из 2738 нуклеотидов были выровнены 2658, что составляет 97,08%.

Также стоит заметить, что линия на графике (рис. 1) имеет наклон и убывает. Это может говорить, что контиг был прочитан в обратном направлении.

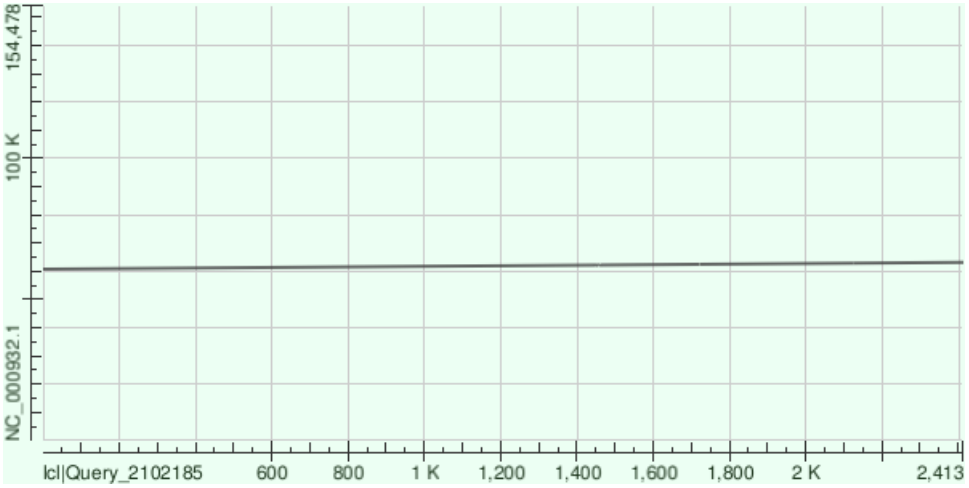

Рис. 2. Dot Plot Conting 2383

Характеристики:

Score	Expect	Identities	Gaps	Strand
4414 bits(2390)	0.0	2407/2414(99%)	6/2414(0%)	Plus/Plus

Характеристики выравнивания показывают, что контиг длинной 2383 нуклеотида является почти полностью идентичным фрагменту хлоропластной хромосомы.

Соответствие найдено с 60768 по 63176 нуклеотид хлоропластной хромосомы.

Из 2383 нуклеотидов были выровнены 2380, что составляет 99,87%.

Из графика (рис. 2) видно, что линия возрастает. Из этого следует, что контиг был прочитан в прямом направлении.

Рис. 3. Dot Plot Conting 2120

Характеристики:

Score	Expect	Identities	Gaps	Strand
3856 bits(2088)	0.0	2134/2154(99%)	12/2154(0%)	Plus/Minus

Как и в случае с двумя предыдущими выравниваниями, данный контиг является почти полностью идентичным со своим фрагментом.

Контиг соответствует хлоропластной хромосоме с 12842 по 14987 нуклеотид.

Из 2120 нуклеотидов были выровнены 2076, что составляет 97,92%.

На графике (рис. 3) можно заметить, что линия как и в первом случае убывает, а значит контиг также прочитан в обратном направлении.

Приложение

```
    contigs_dict = {}
    with open("contigs.fa", "r") as file:
        for line in file:
            line = line.strip()
            if not line or not line.startswith('>'):
                continue

            parts = line.split("_")
            if len(parts) >= 6:
                try:
                    contigs_dict[parts[1]] = [int(parts[3]) + 30,
float(parts[5])]

    len_cov = sorted([x for x in contigs_dict.values()],
key=lambda x: x[0], reverse=True)
    if len_cov:
        half_sum = sum(x[0] for x in len_cov) / 2
        summy = 0
        L50 = 0
        while summy < half_sum and L50 < len(len_cov):
            summy += len_cov[L50][0]
            N50 = len_cov[L50][0]
            L50 += 1
        print(N50, L50)
```