

1. Ссылка на загруженные прочтения (*Escherichia coli*)
https://trace.ncbi.nlm.nih.gov/Traces/?view=run_browser&acc=SRR24631089&display=data-access

2. Скачиваем **GCF_000005845.2_ASM584v2_genomic.fna.gz** и помещаем в рабочую директорию, далее:

Перед началом работы надо выполнить:

```
vdb-config --prefetch-to-cwd
gzip -d GCF_000005845.2_ASM584v2_genomic.fna.gz
prefetch SRR24631089
fasterq-dump --split-files SRR24631089
```

Далее bash скрипт:

```
1. fastqc SRR24631089_1.fastq & fastqc SRR24631089_2.fastq (выполняем параллельно)
2. bwa index GCF_000005845.2_ASM584v2_genomic.fna
3. bwa mem GCF_000005845.2_ASM584v2_genomic.fna SRR24631089_1.fastq
   SRR24631089_2.fastq -o out.sam
4. samtools view -b -o view.bam out.sam
5. samtools flagstat view.bam | python parser.py
6. samtools sort -o sort.bam view.bam
7. samtools index sort.bam
8. freebayes -f GCF_000005845.2_ASM584v2_genomic.fna -b sort.bam --vcf res_bash.vcf
```

! На самом деле из этих операций уже можно построить пайплайн в unix системе используя конвейеры bash (оператор |), достаточно перенаправлять поток вывода *i* команды в поток ввода (*i + 1*) команды. Однако, почти все используемые утилиты умею писать в стандартный поток вывода, а вот читать из stdin не умеют, + ограничение накладывает необходимость использования индексных файлов (их все же пришлось бы создавать). !

3. Результат **samtools flagstat view.bam** (который перенаправляется в скрипт parser.py):

```
5379377 + 0 in total (QC-passed reads + QC-failed reads)
0 + 0 secondary
41795 + 0 supplementary
0 + 0 duplicates
4149410 + 0 mapped (77.14% : N/A)
5337582 + 0 paired in sequencing
2668791 + 0 read1
2668791 + 0 read2
4026058 + 0 properly paired (75.43% : N/A)
4058752 + 0 with itself and mate mapped
48863 + 0 singletons (0.92% : N/A)
0 + 0 with mate mapped to a different chr
0 + 0 with mate mapped to a different chr (mapQ>=5)
```

4. Скрипт разбора файлов с этими результатами - **parser.py**:

```
while True:
    line = input()
    if not line:
        break
    if line.count("%") > 0:
```

```

value = int(line[line.index("(") + 1:line.index(".")]
print("OK" if value >= 90 else "Not OK")
exit(0)

```

5. **SRR24631089_1.fastq** и **SRR24631089_2.fastq** весят по 1.41 GB

out.sam весит 2.99 GB

view.bam весит 1.10 GB

Поэтому в репозитории не находятся.

Файл **res.vcf** лежит в репозитории.

6. Установим компилятор языка Go, создадим проект и скачаем SciPipe:

1. `sudo apt install golang`
2. `mkdir pipeline`
3. `cd pipeline`
4. `go mod init pipeline`
5. `go get github.com/scipipe/scipipe`
6. `go run test.go`

7. Код тестового пайплайна **test.go**:

```

package main

import (
    sp "github.com/scipipe/scipipe"
)

func main() {
    wf := sp.NewWorkflow("hello_world", 4)
    hello := wf.NewProc("hello", "echo 'Hello ' > {o:out|.txt}")
    world := wf.NewProc("world", "echo $(cat {i:in}) World > {o:out|.txt}")
    world.In("in").From(hello.Out("out"))
    wf.Run()
}

```

8. Результат работы тестового пайплайна - файл **hello.out.txt.world.out.txt** (фреймворк очень смешно именует файлы...), **лог** работы:

```

AUDIT 2023/05/25 02:21:22 [Workflow:hello_world] Starting workflow (Writing log to
log/scipipe-20230525-022122-hello_world.log)
AUDIT 2023/05/25 02:21:22 [Task:hello] Executing: echo 'Hello ' > hello.out.txt
AUDIT 2023/05/25 02:21:22 [Task:hello] Finished: echo 'Hello ' > hello.out.txt
AUDIT 2023/05/25 02:21:22 [Task:world] Executing: echo $(cat ../hello.out.txt) World >
hello.out.txt.world.out.txt
AUDIT 2023/05/25 02:21:22 [Task:world] Finished: echo $(cat ../hello.out.txt) World >
hello.out.txt.world.out.txt
AUDIT 2023/05/25 02:21:22 [Workflow:hello_world] Finished workflow (Log written to
log/scipipe-20230525-022122-hello_world.log)

```

9. -

- 10.

9. *Опционально описание использованных инструментов для визуального создания пайплайнов (скриншоты)

10. Код пайплайна “оценки качества картирования” на фреймворке

11. Выведенные результаты работы пайплайна на загруженных данных в отдельном файле

12. Лог-файлы работы пайплайна на загруженных данных

13. Визуализацию пайплайна в виде графического файла

14. Описание использованного способа визуализации и отличия полученной визуализации от блок-схемы алгоритма в свободной форме