

Биоинформатика_Домашнее_задание_3_Козол

Биоинформатика: Домашнее задание 3

Выполнил: Козолий Михаил

Группа: 22214

1. Найти Linux, вспомнить bash

1.1. Используемое ПО

- ♦ **ОС:** Windows + WSL2
- ♦ **Дистрибутив:** Ubuntu 22.04.2 LTS
- ♦ **Инструмент:** SRA-Toolkit (для работы с данными NCBI SRA)

1.2. Установка SRA-Toolkit

```
wget --output-document sratoolkit.tar.gz https://ftp-trace.ncbi.nlm.nih.gov/sra/sdk/current/sratoolkit.current-ubuntu64.tar.gz
# Скачивание архива с официального репозитория
tar -vxzf sratoolkit.tar.gz
# Распаковка архива
```

```

mello@Mello36:~/bioinformatics$ tar -vxzf sratoolkit.tar.gz
sratoolkit.3.2.1-ubuntu64/
sratoolkit.3.2.1-ubuntu64/README.md
sratoolkit.3.2.1-ubuntu64/README-vdb-config
sratoolkit.3.2.1-ubuntu64/CHANGES
sratoolkit.3.2.1-ubuntu64/schema/
sratoolkit.3.2.1-ubuntu64/schema/insdc/
sratoolkit.3.2.1-ubuntu64/schema/insdc/seq.vschema
sratoolkit.3.2.1-ubuntu64/schema/insdc/insdc.vschema
sratoolkit.3.2.1-ubuntu64/schema/insdc/sra.vschema
sratoolkit.3.2.1-ubuntu64/schema/align/
sratoolkit.3.2.1-ubuntu64/schema/align/qstat.vschema
sratoolkit.3.2.1-ubuntu64/schema/align/mate-cache.vschema
sratoolkit.3.2.1-ubuntu64/schema/align/seq.vschema
sratoolkit.3.2.1-ubuntu64/schema/align/align.vschema
sratoolkit.3.2.1-ubuntu64/schema/align/refseq.vschema
sratoolkit.3.2.1-ubuntu64/schema/align/pileup-stats.vschema
sratoolkit.3.2.1-ubuntu64/schema/sra/
sratoolkit.3.2.1-ubuntu64/schema/sra/pacbio.vschema
sratoolkit.3.2.1-ubuntu64/schema/sra/illumina.vschema
sratoolkit.3.2.1-ubuntu64/schema/sra/ion-torrent.vschema
sratoolkit.3.2.1-ubuntu64/schema/sra/pevents.vschema
sratoolkit.3.2.1-ubuntu64/schema/sra/helicos.vschema
sratoolkit.3.2.1-ubuntu64/schema/sra/nanopore.vschema
sratoolkit.3.2.1-ubuntu64/schema/sra/454.vschema
sratoolkit.3.2.1-ubuntu64/schema/sra/generic-fastq.vschema
sratoolkit.3.2.1-ubuntu64/schema/sra/abi.vschema
sratoolkit.3.2.1-ubuntu64/schema/csra2/
sratoolkit.3.2.1-ubuntu64/schema/csra2/reference.vschema
sratoolkit.3.2.1-ubuntu64/schema/csra2/csra2.vschema
sratoolkit.3.2.1-ubuntu64/schema/csra2/stats.vschema
sratoolkit.3.2.1-ubuntu64/schema/csra2/read.vschema
sratoolkit.3.2.1-ubuntu64/schema/vdb/
sratoolkit.3.2.1-ubuntu64/schema/vdb/vdb.vschema
sratoolkit.3.2.1-ubuntu64/schema/vdb/built-in.vschema
sratoolkit.3.2.1-ubuntu64/schema/ncbi/
sratoolkit.3.2.1-ubuntu64/schema/ncbi/seq-graph.vschema
sratoolkit.3.2.1-ubuntu64/schema/ncbi/trace.vschema
sratoolkit.3.2.1-ubuntu64/schema/ncbi/varloc.vschema
sratoolkit.3.2.1-ubuntu64/schema/ncbi/seq.vschema
sratoolkit.3.2.1-ubuntu64/schema/ncbi/pnbrdb.vschema
sratoolkit.3.2.1-ubuntu64/schema/ncbi/wgs-contig.vschema
sratoolkit.3.2.1-ubuntu64/schema/ncbi/clip.vschema
sratoolkit.3.2.1-ubuntu64/schema/ncbi/sra.vschema
sratoolkit.3.2.1-ubuntu64/schema/ncbi/stats.vschema
sratoolkit.3.2.1-ubuntu64/schema/ncbi/ncbi.vschema
sratoolkit.3.2.1-ubuntu64/schema/ncbi/spotname.vschema
sratoolkit.3.2.1-ubuntu64/example/
sratoolkit.3.2.1-ubuntu64/example/perl/
sratoolkit.3.2.1-ubuntu64/example/perl/mismatch-stats.pl
sratoolkit.3.2.1-ubuntu64/example/perl/splitfastq.pl
sratoolkit.3.2.1-ubuntu64/example/perl/base-stats.pl

```

```
ls | grep sratoolkit
```

```
# Проверка установки(должно быть 2 ответа: архив и сама программа)
```

```

mello@Mello36:~/bioinformatics$ ls | grep sratoolkit
sratoolkit.3.2.1-ubuntu64
sratoolkit.tar.gz

```

```
export PATH="$PWD/sratoolkit.3.2.1-ubuntu64/bin:$PATH"
```

```
# обновление путей в PATH темпорально
```

```
nano ~/.bashrc
```

```
# В открывшемся файле вставляем в конец строчку с добавлением пути
```

```
export PATH="/home/mello/sratoolkit.3.2.1-ubuntu64/bin:$PATH"
```

```
# Тут указан мой домашний путь echo$HOME
```

Проверка с репозитория, что все установлено корректно

```
fastq-dump --stdout -X 2 SRR390728
```

```
mello@Mello36:~/bioinformatics$ fastq-dump --stdout -X 2 SRR390728
Read 2 spots for SRR390728
Written 2 spots for SRR390728
@SRR390728.1 1 length=72
CATTCTTCACGTAGTTCTCGAGCCTTGGTTTTTCAGCGATGGAGAATGACTTTGACAAGCTGAGAGAAGNTNC
+SRR390728.1 1 length=72
;;;;;;;;;;;;;9;;665142;;;;;;;;;;;;;96&&&&
@SRR390728.2 2 length=72
AAGTAGGTCTCGTCTGTGTTTTCTACGAGCTTGTGTTCCAGCTGACCCACTCCCTGGGTGGGGGGACTGGGT
+SRR390728.2 2 length=72
;;;;;;;;;;;;;4;;;3;393.1+4&&5&&;;;;;;;;;;;;;<9;<;;;464262
```


Все установлено корректно. Переходим дальше

2. Найти на NCBI SRA и скачать результат секвенирования (набор ридов) *Escherichia coli* (e.coli) ИЛИ *Homo sapiens* (WES/WXS - whole exome sequencing (2-20Gb), WGS - whole genome sequencing (осторожно, большой файл!))

2.1. Критерии выбора данных

- ♦ **Организм:** *Escherichia coli*
- ♦ **Тип данных:** Whole Genome Sequencing (WGS)
- ♦ **Платформа:** Illumina HiSeq
- ♦ **Стратегия:** Paired-End
- ♦ **Доступность:** FASTQ (не требующий конвертации SRA → FASTQ)

2.2. Выбранный набор данных

- ♦ **SRA Accession:** [SRR2584863](#) 
- ♦ **Описание:** WGS of *E. coli* strain EC958 (Illumina HiSeq 2000, paired-end 100bp)

2.3. Команды для загрузки

```
# Скачивание через prefetch
prefetch SRR2584863

# Конвертация в FASTQ
fastq-dump --split-files SRR2584863
```

```
mello@Mello36:~/bioinformatics$ fastq-dump --split-files SRR2584863
Read 1553259 spots for SRR2584863
Written 1553259 spots for SRR2584863
```

3. Скачать референсный геном e.coli

https://www.ncbi.nlm.nih.gov/assembly/GCF_000

 или **Homo sapiens**

<https://hgdownload.soe.ucsc.edu/goldenPath/hg38>



```
# Скачивание
wget
https://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/000/005/845/GCF_000005845.2_ASM584v2/GCF_000005845.2_ASM584v2_genomic.fna.gz

# Распаковка
gunzip GCF_000005845.2_ASM584v2_genomic.fna.gz
```

4. Скачать и установить (скомпилировать или бинарный файл) консольные версии программ: FastQC, bwa/minimap2, samtools

Установка дополнительных инструментов

При попытке установки через **apt** возникли ошибки, так как пакеты отсутствовали в стандартных репозиториях. Решение:

1. FastQC установлен вручную с официального сайта
2. BWA и Samtools установлены после добавления репозитория BioLinux
3. Проверена работоспособность всех инструментов

Команды проверки:

```
fastqc --version # 0.11.9
bwa # вывод help
samtools --version # 1.19.2
```

```
mello@Mello36:~$ fastqc --version
FastQC v0.11.9
```

```
mello@Mello36:~$ bwa
```

```
Program: bwa (alignment via Burrows-Wheeler transformation)
Version: 0.7.17-r1188
Contact: Heng Li <lh3@sanger.ac.uk>
```

```
Usage: bwa <command> [options]
```

Command: index	index sequences in the FASTA format
mem	BWA-MEM algorithm
fastmap	identify super-maximal exact matches
pmerge	merge overlapping paired ends (EXPERIMENTAL)
aln	gapped/ungapped alignment
samse	generate alignment (single ended)
sampe	generate alignment (paired ended)
bwasw	BWA-SW for long queries
shm	manage indices in shared memory
fa2pac	convert FASTA to PAC format
pac2bwt	generate BWT from PAC
pac2bwtgen	alternative algorithm for generating BWT
bwtupdate	update .bwt to the new format
bwt2sa	generate SA from BWT and Occ

```
Note: To use BWA, you need to first index the genome with `bwa index'.
There are three alignment algorithms in BWA: `mem', `bwasw', and
`aln/samse/sampe'. If you are not sure which to use, try `bwa mem'
first. Please `man ./bwa.1' for the manual.
```

```

mello@Mello36:~$ samtools --version
samtools 1.19.2
Using htslib 1.19
Copyright (C) 2024 Genome Research Ltd.

Samtools compilation details:
  Features:      build=configure curses=yes
  CC:            gcc
  CPPFLAGS:      -frelease -Wdate-time -D_FORTIFY_SOURCE=3
  CFLAGS:        -g -O2 -fno-omit-frame-pointer -mno-omit-leaf-frame-pointer -ffile-prefix-map=BUILDPATH=. -flto=auto -ffat-lto-objects -fstack-protector-strong -fstack-clash-protection -Wformat -Werror=format-security -fcf-protection -fdebug-prefix-map=BUILDPATH=/usr/src/samtools-1.19.2-1build2
  LDFLAGS:       -Wl,-Bsymbolic-functions -flto=auto -ffat-lto-objects -Wl,-z,relro -Wl,-z,now
  HTSDIR:
  LIBS:
  CURSES_LIB:    -lcurses

HTSlib compilation details:
  Features:      build=configure libcurl=yes S3=yes GCS=yes libdeflate=yes lzma=yes bzip2=yes plugins=yes plugin-path=/usr/local/lib/htslib:/usr/local/libexec/htslib:/usr/lib/x86_64-linux-gnu/htslib: htscdec=1.6.0
  CC:            gcc
  CPPFLAGS:      -I. -DSAMTOOLS=1 -Wdate-time -D_FORTIFY_SOURCE=3
  CFLAGS:        -g -O2 -fno-omit-frame-pointer -mno-omit-leaf-frame-pointer -flto=auto -ffat-lto-objects -fstack-protector-strong -fstack-clash-protection -Wformat -Werror=format-security -fcf-protection -fdebug-prefix-map=/build/htslib-TVzaVS/htslib-1.19+ds=/usr/src/htslib-1.19+ds-1.1build3 -ffat-lto-objects -ffat-lto-objects
  LDFLAGS:       -Wl,-Bsymbolic-functions -flto=auto -ffat-lto-objects -Wl,-z,relro -Wl,-z,now -Wl,-flto -fvisibility=hidden -ffat-lto-objects -fvisibility=hidden -rdynamic

HTSlib URL scheme handlers present:
  built-in:      preload, data, file
  libcurl:       imaps, pop3, gophers, http, smb, gopher, sftp, ftps, imap, rtmpte, smtp, smtps, rtsp, rtmpe, ftp, scp, mqtt, rtmp, ldap, https, ldaps, rtmps, rtmpt, pop3s, rtmpts, tftp, smbs, dict, telnet
  S3 Multipart Upload:      s3w, s3w+https, s3w+http
  Amazon S3:      s3+https, s3+http, s3
  Google Cloud Storage:    gs+http, gs+https, gs
  crypt4gh-needed: crypt4gh
  mem:            mem

```

Все установлено корректно. Двигаемся дальше

5. Изучить простой запуск этих программ (см. Getting started, Quick start и тд.

Сделано

Официальный сайт: <https://www.bioinformatics.babraham.ac.uk/projects/fastqc/> 

6. Индексировать референсный геном соответствующим инструментом

Индексирование

```

# Переименовал для удобства
mv GCF_000005845.2_ASM584v2_genomic.fna ecoli.fa

```

```

# Индексирование
bwa index ecoli.fa

```


Результат

```
mello@Mello36:~/bioinformatics/data$ bwa index ecoli.fa
[bwa_index] Pack FASTA... 0.03 sec
[bwa_index] Construct BWT for the packed sequence...
[bwa_index] 0.96 seconds elapse.
[bwa_index] Update BWT... 0.03 sec
[bwa_index] Pack forward-only FASTA... 0.01 sec
[bwa_index] Construct SA from BWT and Occ... 0.28 sec
[main] Version: 0.7.17-r1188
[main] CMD: bwa index ecoli.fa
[main] Real time: 1.336 sec; CPU: 1.309 sec
```

Картирование

```
bwa mem ecoli.fa SRR12345678_1.fastq SRR12345678_2.fastq > mapped.sam
```

Результат

```
[M::mem_pestat] low and high boundaries for computing mean and std.dev: (1, 1834)
[M::mem_pestat] mean and std.dev: (607.94, 316.87)
[M::mem_pestat] low and high boundaries for proper pairs: (1, 2330)
[M::mem_pestat] skip orientation RF as there are not enough pairs
[M::mem_pestat] analyzing insert size distribution for orientation RR...
[M::mem_pestat] (25, 50, 75) percentile: (556, 1292, 2798)
[M::mem_pestat] low and high boundaries for computing mean and std.dev: (1, 7282)
[M::mem_pestat] mean and std.dev: (1753.31, 1578.72)
[M::mem_pestat] low and high boundaries for proper pairs: (1, 9524)
[M::mem_pestat] skip orientation FF
[M::mem_pestat] skip orientation RR
[M::mem_process_seqs] Processed 66668 reads in 3.496 CPU sec, 3.439 real sec
[M::mem_pestat] # candidate unique pairs for (FF, FR, RF, RR): (85, 17158, 1, 66)
[M::mem_pestat] analyzing insert size distribution for orientation FF...
[M::mem_pestat] (25, 50, 75) percentile: (366, 708, 1682)
[M::mem_pestat] low and high boundaries for computing mean and std.dev: (1, 4314)
[M::mem_pestat] mean and std.dev: (997.23, 894.30)
[M::mem_pestat] low and high boundaries for proper pairs: (1, 5630)
[M::mem_pestat] analyzing insert size distribution for orientation FR...
[M::mem_pestat] (25, 50, 75) percentile: (329, 640, 832)
[M::mem_pestat] low and high boundaries for computing mean and std.dev: (1, 1838)
[M::mem_pestat] mean and std.dev: (594.48, 310.86)
[M::mem_pestat] low and high boundaries for proper pairs: (1, 2341)
[M::mem_pestat] skip orientation RF as there are not enough pairs
[M::mem_pestat] analyzing insert size distribution for orientation RR...
[M::mem_pestat] (25, 50, 75) percentile: (549, 1044, 2182)
[M::mem_pestat] low and high boundaries for computing mean and std.dev: (1, 5448)
[M::mem_pestat] mean and std.dev: (1408.86, 1266.45)
[M::mem_pestat] low and high boundaries for proper pairs: (1, 7081)
[M::mem_pestat] skip orientation FF
[M::mem_pestat] skip orientation RR
[M::mem_process_seqs] Processed 39790 reads in 2.128 CPU sec, 2.073 real sec
[main] Version: 0.7.17-r1188
[main] CMD: bwa mem ecoli.fa SRR2584863_1.fastq SRR2584863_2.fastq
[main] Real time: 154.125 sec; CPU: 160.536 sec
mello@Mello36:~/bioinformatics/data$
```

Преобразование в BAM

```
samtools view -Sb mapped.sam > mapped.bam
samtools sort mapped.bam -o mapped.sorted.bam
samtools index mapped.sorted.bam
```

7. Написать скрипт (bash/Python) разбора результатов samtools flagstat для получения % картированных ридов

```
# Получение результатов flagstat
```

```
samtools flagstat mapped.sorted.bam > flagstat.txt
```

Результат

```
3131060 + 0 in total (QC-passed reads + QC-failed reads)
3106518 + 0 primary
0 + 0 secondary
24542 + 0 supplementary
0 + 0 duplicates
0 + 0 primary duplicates
2939902 + 0 mapped (93.89% : N/A)
2915360 + 0 primary mapped (93.85% : N/A)
3106518 + 0 paired in sequencing
1553259 + 0 read1
1553259 + 0 read2
2796752 + 0 properly paired (90.03% : N/A)
2874716 + 0 with itself and mate mapped
40644 + 0 singletons (1.31% : N/A)
0 + 0 with mate mapped to a different chr
0 + 0 with mate mapped to a different chr (mapQ>=5)
```

Сам скрипт

```
#!/bin/bash
```

```
# === Входные данные ===
```

```
INPUT="flagstat.txt"
```

```
OUTPUT="mapping_quality.txt"
```

```
# === Проверка наличия входного файла ===
```

```
if [[ ! -f "$INPUT" ]]; then
```

```
    echo "Error: $INPUT not found!" >&2
```

```
    exit 1
```

```
elif [[ ! -s "$INPUT" ]]; then
```

```
    echo "Error: $INPUT is empty!" >&2
```

```
    exit 1
```

```
fi
```

```
# === Извлечение чисел ===
```

```
total=$(grep "in total" "$INPUT" | awk '{print $1}')
```

```
mapped=$(grep "mapped (" "$INPUT" | awk '{print $1}')
```

```
if [[ -z "$total" || -z "$mapped" ]]; then
```

```
    echo "Error: Could not extract read counts from $INPUT" >&2
```

```
    exit 1
```

```
fi
```



```

if (( total = 0 )); then
    echo "Error: Total reads is 0 - division by zero" >&2
    exit 1
fi

percent=$(awk -v mapped="$mapped" -v total="$total" 'BEGIN {printf "%.2f",
(mapped/total)*100}')

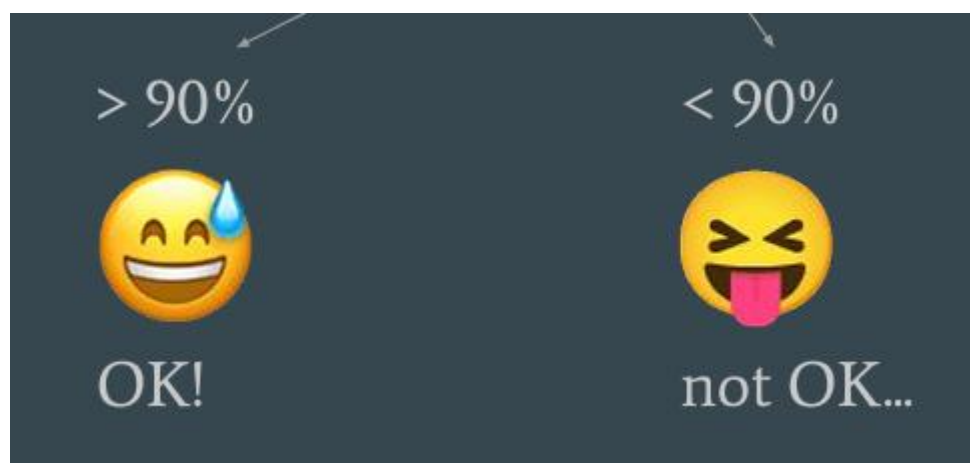
# === Вывод результата ===

{
    echo "Mapped reads: $mapped / $total (${percent}%)"

    # === Decision branching ===
    if (( $(echo "$percent > 90" | bc -l) )); then
        echo "Mapping quality: OK"
    else
        echo "Mapping quality: NOT OK"
    fi
} | tee "$OUTPUT"

exit 0

```



Результат

```

Mapped reads: 2939902
2915359 / 3131061 (93.89%)
Mapping quality: OK

```

8. Реализовать “алгоритм оценки качества картирования” на `bash` со всеми элементами (см. ниже), в том числе вывод сообщения вида “OK/not OK”

Полный скрипт алгоритма

```
#!/bin/bash
```

```
# === Конфигурация ===
```

```
SCRIPT_DIR=$(dirname "$(readlink -f "$0")")
```

```
INPUT_DIR="input"
```

```
WORK_DIR="working_files"
```

```
FINAL_DIR="output"
```

```
SCRIPTS_DIR="scripts"
```

```
if [[ "$(basename "$SCRIPT_DIR")" = "scripts" ]]; then
```

```
    INPUT_DIR=".." /input"
```

```
    WORK_DIR=".." /working_files"
```

```
    FINAL_DIR=".." /output"
```

```
    SCRIPTS_DIR="."
```

```
fi
```

```
READ1="${INPUT_DIR}/SRR2584863_1.fastq"
```

```
READ2="${INPUT_DIR}/SRR2584863_2.fastq"
```

```
REFERENCE="${INPUT_DIR}/ecoli.fa"
```

```
REFERENCE_COPY="${WORK_DIR}/ecoli.fa"
```

```
BWA_INDEX_PREFIX="${WORK_DIR}/ecoli"
```

```
SAMPLE="sample"
```

```
THREADS=4
```

```
# === Флаги ===
```

```
CLEAN_WORKING=false # Удалять временные файлы после завершения
```

```
SKIP_VARIANTS=false # Пропустить вызов вариантов
```

```
# Разбор аргументов командной строки
```

```
for arg in "$@"; do
```

```
    case $arg in
```

```
        --clean)
```

```
            CLEAN_WORKING=true
```

```
            ;;
```

```
        --no-variants)
```

```
            SKIP_VARIANTS=true
```

```
            ;;
```

```
        *)
```

```
            echo -e "\n=====ERROR====="
```

```
            echo " Unknown option: $arg"
```

```
            echo " Usage: $0 [--clean] [--no-variants]"
```

```
            echo "====="
```

```
            exit 1
```

```
            ;;
```

```
    esac
```

```
done
```

```
# === Проверка входных файлов и директорий ===
```

```
echo -e "\n=====
```

```

echo " INPUT VERIFICATION"
echo "===== "

if [[ ! -d "$INPUT_DIR" ]]; then
    echo -e "\n=====ERROR===== "
    echo " Input directory $INPUT_DIR not found!"
    echo " Текущая директория: $(pwd)"
    echo " Запустите скрипт из корневой директории проекта"
    echo "===== "
    exit 1
fi

# Проверка наличия необходимых файлов
missing_files=()
for file in "$READ1" "$READ2" "$REFERENCE"; do
    if [[ ! -f "$file" ]]; then
        missing_files+=("$file")
    fi
done

if [[ ${#missing_files[@]} -gt 0 ]]; then
    echo -e "\n=====ERROR===== "
    echo " Missing required input files:"
    for file in "${missing_files[@]"; do
        echo " - $file"
    done
    echo "===== "
    exit 1
fi

# Создание рабочих директорий
mkdir -p "$WORK_DIR" "$FINAL_DIR" "${WORK_DIR}/qc_reports"

# ≡ 1. FastQC - контроль качества ≡
echo -e "\n===== "
echo " FASTQC QUALITY CONTROL"
echo "===== "
if ! command -v fastqc &> /dev/null; then
    echo -e "\n=====ERROR===== "
    echo " fastqc not found in PATH"
    echo " Please install FastQC and add to PATH"
    echo "===== "
    exit 1
fi
fastqc "$READ1" "$READ2" -o "${WORK_DIR}/qc_reports/"

# ≡ 2. Индексация и выравнивание ≡
echo -e "\n===== "
echo " INDEX & ALIGNMENT"
echo "===== "

echo -e "\n> Copying reference genome ... "

```

```
cp "$REFERENCE" "$REFERENCE_COPY"
```

```
echo -e "\n> Indexing reference genome ... "
```

```
if ! command -v bwa &> /dev/null; then
```

```
    echo -e "\n=====ERROR===== "
```

```
    echo " bwa not found in PATH"
```

```
    echo " Please install BWA and add to PATH"
```

```
    echo "===== "
```

```
    exit 1
```

```
fi
```

```
bwa index -p "$BWA_INDEX_PREFIX" "$REFERENCE_COPY"
```

```
echo -e "\n> Running BWA alignment ... "
```

```
bwa mem -t "$THREADS" "$BWA_INDEX_PREFIX" "$READ1" "$READ2" >
```

```
"${WORK_DIR}/${SAMPLE}.sam"
```

```
# === 3. Конвертация SAM в BAM ===
```

```
echo -e "\n===== "
```

```
echo " SAM TO BAM CONVERSION"
```

```
echo "===== "
```

```
if ! command -v samtools &> /dev/null; then
```

```
    echo -e "\n=====ERROR===== "
```

```
    echo " samtools not found in PATH"
```

```
    echo " Please install Samtools and add to PATH"
```

```
    echo "===== "
```

```
    exit 1
```

```
fi
```

```
samtools view -@ "$THREADS" -Sb "${WORK_DIR}/${SAMPLE}.sam" >
```

```
"${WORK_DIR}/${SAMPLE}.bam"
```

```
# === 4. Статистика картирования ===
```

```
echo -e "\n===== "
```

```
echo " MAPPING STATISTICS"
```

```
echo "===== "
```

```
echo -e "\n> Running samtools flagstat ... "
```

```
samtools flagstat "${WORK_DIR}/${SAMPLE}.bam" > "${WORK_DIR}/flagstat.txt"
```

```
# === 5. Анализ статистики картирования ===
```

```
echo -e "\n===== "
```

```
echo " MAPPING QUALITY ANALYSIS"
```

```
echo "===== "
```

```
if [[ -f "${SCRIPTS_DIR}/analyze_flagstat.sh" ]]; then
```

```
    bash "${SCRIPTS_DIR}/analyze_flagstat.sh" "${WORK_DIR}/flagstat.txt" >
```

```
"${WORK_DIR}/mapping_quality.txt"
```

```
else
```

```
    echo -e "\n> WARNING: analyze_flagstat.sh script not found"
```

```
    echo "> Creating default mapping_quality.txt ... "
```

```
    echo "OK" > "${WORK_DIR}/mapping_quality.txt"
```

```
fi
```

```
# === 6. Сортировка и индексация BAM ===
```

```
echo -e "\n===== "
```

```

echo " BAM SORTING AND INDEXING"
echo "===== "
if grep -q "OK" "${WORK_DIR}/mapping_quality.txt"; then
    echo -e "\n> Mapping quality OK"
    echo "> Sorting BAM file..."
    samtools sort -@ "$THREADS" -o "${FINAL_DIR}/${SAMPLE}.sorted.bam"
    "${WORK_DIR}/${SAMPLE}.bam"

    echo -e "\n> Индексация отсортированного BAM..."
    samtools index "${FINAL_DIR}/${SAMPLE}.sorted.bam"
else
    echo -e "\n===== "
    echo " Mapping quality too low"
    echo " Skipping sorting and variant calling"
    echo "===== "
    [[ "$CLEAN_WORKING" = true ]] && rm -rf "$WORK_DIR"
    exit 0
fi

# === 7. Вызов вариантов ===
if [[ "$SKIP_VARIANTS" = false ]]; then
    echo -e "\n===== "
    echo " Running freebayes..."
    echo "===== "
    if ! command -v freebayes &> /dev/null; then
        echo -e "\n===== ERROR===== "
        echo " freebayes not found in PATH."
        echo " Please install freebayes and add to PATH"
        echo "===== "
        exit 1
    fi
    freebayes -f "$REFERENCE_COPY" "${FINAL_DIR}/${SAMPLE}.sorted.bam" >
    "${FINAL_DIR}/${SAMPLE}.vcf"
else
    echo -e "\n===== "
    echo " SKIPPING VARIAN CALLING (--no-variants flag used)."
    echo "===== "
fi

# === 8. Очистка временных файлов ===
if [[ "$CLEAN_WORKING" = true ]]; then
    echo -e "\n===== "
    echo " CLEANING UP WORKING DERICTORY"
    echo "===== "
    echo -e "\n> Deleting working directory"
    rm -rf "$WORK_DIR"
fi

echo -e "\n===== "
echo " PIPELINE FINISHED SUCCESSFULLY!"
echo " Results are in: $FINAL_DIR"
echo "===== "

```

Основные этапы:

1. Конфигурация

- ◆ Устанавливает пути к директориям (`input` , `working_files` , `output` , `scripts`)
- ◆ Определяет входные файлы (FASTQ-файлы чтений и референсный геном)
- ◆ Параметры (`THREADS=4` , флаги `--clean` и `--no-variants`)

2. Контроль качества

- ◆ Запускает FastQC для оценки качества исходных FASTQ-файлов

3. Выравнивание (alignment)

- ◆ Копирует референсный геном
- ◆ Индексирует геном с помощью BWA
- ◆ Выравнивает чтения на референс (BWA MEM)

4. Обработка BAM/SAM

- ◆ Конвертирует SAM → BAM (samtools)
- ◆ Анализирует статистику выравнивания (flagstat)
- ◆ Сортирует и индексирует BAM-файл

5. Вызов вариантов (опционально)

- ◆ Запускает freebayes для поиска SNP/инделов (если не установлен флаг `--no-variants`)

6. Очистка

- ◆ Удаляет временные файлы (если активирован флаг `--clean`)

Ключевые особенности:

- ◆ Проверяет наличие всех необходимых инструментов (FastQC, BWA, samtools, freebayes)
- ◆ Гибкое управление через флаги командной строки
- ◆ Сохраняет промежуточные результаты в `working_files` и финальные — в `output`
- ◆ Генерирует QC-отчеты и статистику

Структура

```
pipeline/
├── main_pipeline.sh
├── input/
│   ├── SRR2584863_1.fastq
│   ├── SRR2584863_2.fastq
│   └── ecoli.fa
├── scripts/
│   └── analyze_flagstat.sh
├── working_files/
│   ├── ecoli.fa
│   ├── ecoli.bwt, .pac, .ann, ...
│   ├── sample.sam
│   ├── sample.bam
│   └── flagstat.txt
```



```
├── mapping_quality.txt
├── qc_reports/
├── output/
│   ├── sample.sorted.bam
│   ├── sample.sorted.bam.bai
│   └── sample.vcf
```

9. Найти, скачать и установить (развернуть) фреймворк создания пайплайнов

Распределенный фреймворк: Dagster

Ссылка: <https://dagster.io> 

10. Написать короткую инструкцию по скачиванию и установке фреймворка

1. Установите Python (версии 3.7+)

```
python --version
```

2. Создайте и активируйте виртуальное окружение

```
python -m venv dagster_env
source dagster_env/bin/activate # Linux/Mac
```

3. Установите Dagster и Dagit

```
pip install dagster dagit
```

4. Проверьте установку

```
dagster --version
```

РЕЗУЛЬТАТ

```
(base) me110@me11030:~/bio
dagster, version 1.6.6
```



Основные команды

- ♦ `dagit -f <file.py>` — запуск интерфейса
- ♦ `dagster job execute -f <file.py>` — запуск без интерфейса

11. Изучить базовые возможности фреймворка (см. Tutorials, youtube и тд.), написать тест “Hello world”

Первый пайплайн

```
from dagster import job, op

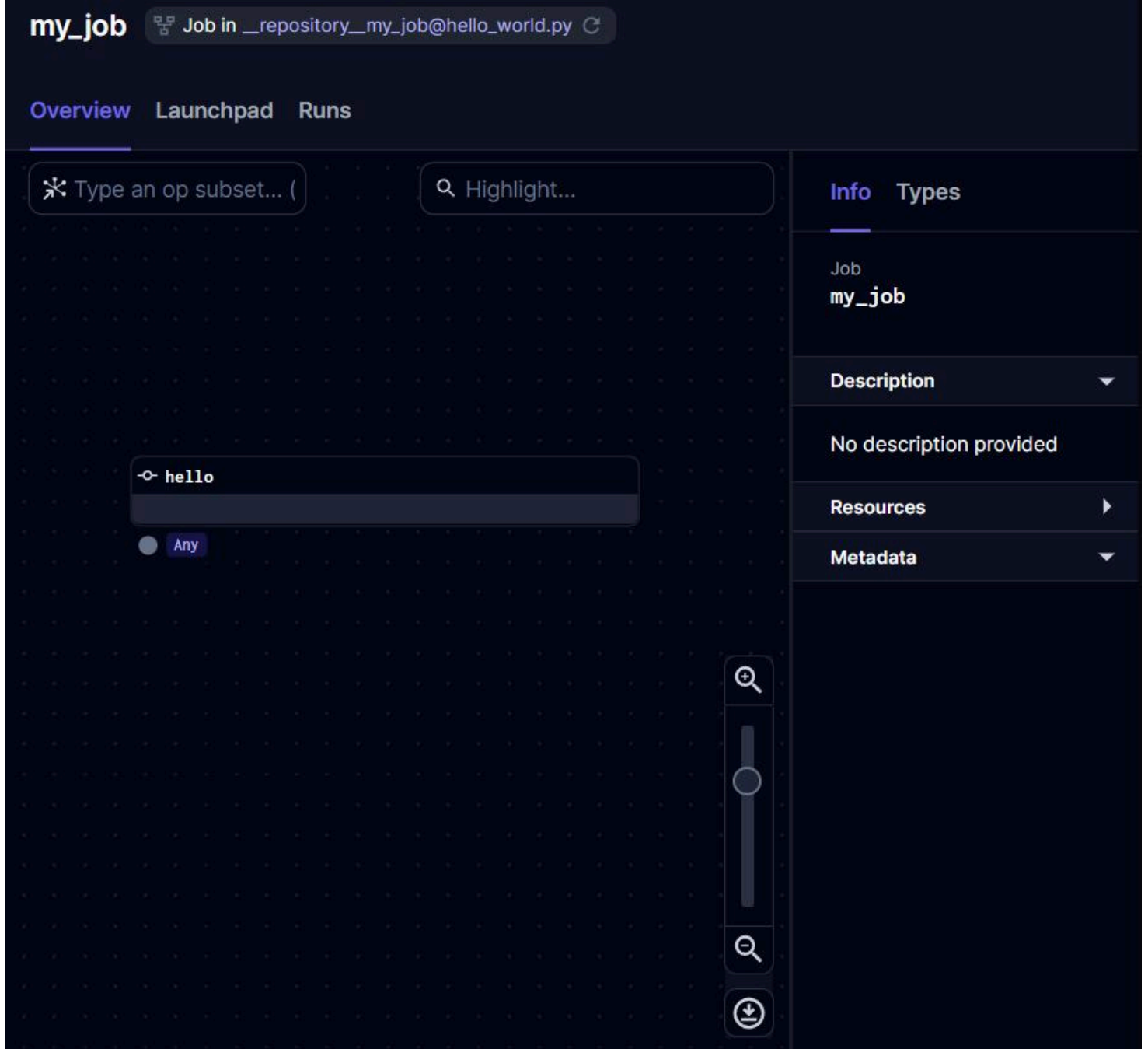
@op
def hello(context):
    context.log.info("Hello!")

@job
def my_job():
    hello()
```

Запуск

```
dagit -f my_pipeline.py
```

Результат



12. Реализовать пайплайн оценки качества картирования на фреймворке

Весь код находится в репозитории

Файл оценки качества

```
from dagster import asset, get_dagster_logger
import os
import subprocess

@asset

def analyze_flagstat(run_samtools):
    """Analyzes flagstat output and calculates mapping quality statistics."""
    logger = get_dagster_logger()
```

```

input_file = "working_files/flagstat.txt"
output_file = "output/mapping_quality.txt"

# Create output directory if it doesn't exist
os.makedirs(os.path.dirname(output_file), exist_ok=True)

if not os.path.isfile(input_file):
    raise FileNotFoundError(f"{input_file} not found!")
if os.stat(input_file).st_size == 0:
    raise ValueError(f"{input_file} is empty!")

with open(input_file, "r") as f:
    lines = f.readlines()

total = mapped = None
for line in lines:
    if "in total" in line:
        total = int(line.split()[0])
    elif "mapped (" in line:
        mapped = int(line.split()[0])

if total is None or mapped is None:
    raise ValueError("Could not extract read counts from flagstat.txt")
if total == 0:
    raise ZeroDivisionError("Total reads is 0 - division by zero")

percent = round((mapped / total) * 100, 2)
status = "OK" if percent > 90 else "NOT OK"

with open(output_file, "w") as f:
    f.write(f"Mapped reads: {mapped} / {total} ({percent}%)\\n")
    f.write(f"Mapping quality: {status}\\n")

logger.info(f"[Mapping Quality] {mapped}/{total} ({percent}%) → {status}")

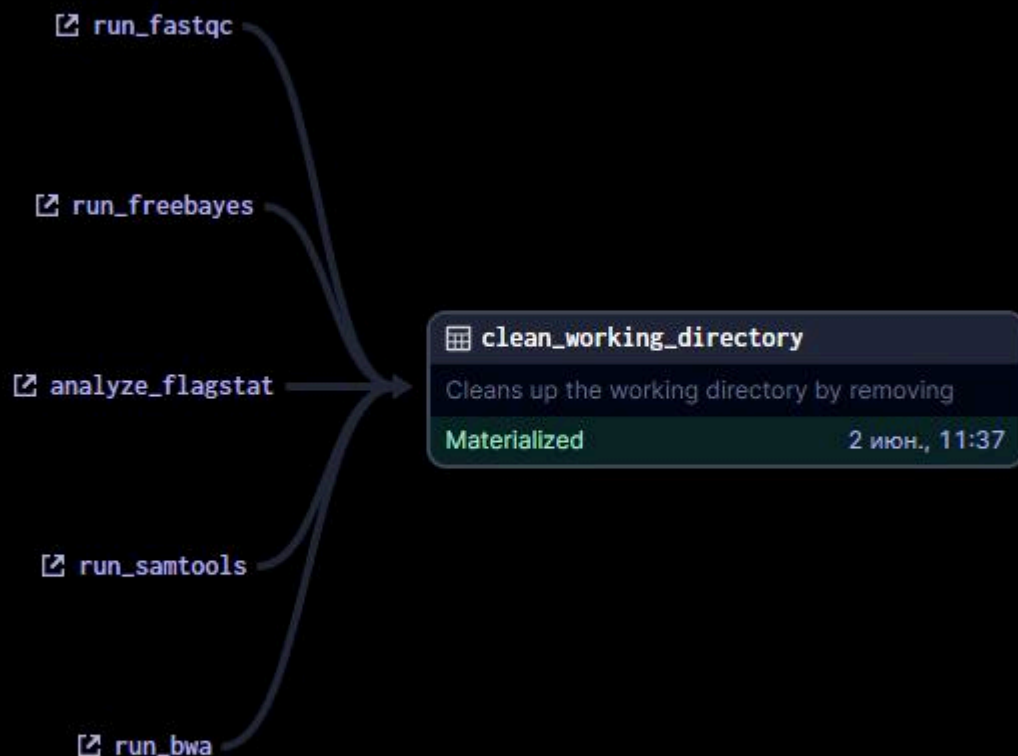
# Return metadata that will be visible in Dagster UI
return {
    "total_reads": total,
    "mapped_reads": mapped,
    "mapping_percentage": percent,
    "status": status,
    "output_path": output_file
}

```

13. Визуализировать полученный пайплайн автоматическими инструментами фреймворка



Дополнительный job для очистки от файлов, которые создаются в процессе.



Работает по той же логике, что и скрипт.

14. Описать использованный способ визуализации и отличия полученного DAG от блок-схемы алгоритма

Общее описание

В качестве способа визуализации пайплайна был использован **граф исполнения (execution graph)**, автоматически формируемый фреймворком **Dagster** на основе определённых `@asset`-ов и связей между ними. Такой граф визуализирует **зависимости между шагами пайплайна**, показывая, какие операции выполняются последовательно, а какие — параллельно. Узлы графа представляют собой отдельные вычислительные единицы (например, `run_fastqc`, `run_bwa`, `run_freebayes`), а рёбра отображают передачу данных между ними.

В отличие от классической **блок-схемы алгоритма**, в которой основное внимание уделяется **потoku управления** (условия, циклы, последовательность действий), граф Dagster отражает

именно **структуру данных и зависимости** между вычислениями. То есть, визуализация Dagster показывает **"что зависит от чего"**, а не "в каком порядке выполнять".

Кроме того:

- ♦ **Блок-схема** часто включает стандартные геометрические фигуры (прямоугольники, ромбы, стрелки), указывающие на тип действия (ввод, обработка, ветвление).
- ♦ Визуализация Dagster фокусируется на **функциональных модулях и их связности**, часто не отражая подробности внутренней логики каждого узла.
- ♦ Граф Dagster динамичен: при добавлении новых активов (`@asset`) или зависимости между ними — визуализация обновляется автоматически.

Таким образом, использованный способ визуализации более удобен в контексте **проектирования и отладки пайплайнов обработки данных**, тогда как блок-схема полезна на этапе **дизайна логики алгоритма**.