



Tópico 03 - Manipulação de Arquivos Biológicos (FASTA/FASTQ) e Comandos Essenciais (1)



1. Entendendo os formatos FASTA e FASTQ

FASTA

Formato usado para sequências (DNA, RNA, proteínas).

Estrutura:

```
>nome_da_sequencia  
ATGCATGCATGCATGC
```

FASTQ

Formato usado para dados brutos de sequenciamento.

Cada *read* ocupa 4 linhas:

```
@Nome_da_read  
ATGCTAGCTAGC  
+  
FFJJFJFJAFJA
```



2. Visualizando arquivos no terminal

Visualizar as primeiras linhas:

```
head arquivo.fasta  
head arquivo.fastq
```

As últimas:

```
tail arquivo.fastq
```

Visualizar com navegação interativa:

```
less arquivo.fastq
```

Dentro do **less**:

- **q** = sair

3. Contando linhas, sequências e reads

Um FASTQ tem **4 linhas por read**:

```
wc -l arquivo.fastq
```

Para contar reads:

```
wc -l arquivo.fastq | awk '{print $1/4 " reads"}'
```

```
zcat arquivo.fastq.gz | wc -l | awk '{print $1/4 " reads"}'
```

Para contar quantas sequências FASTA:

```
grep -c "^>" arquivo.fasta
```

4. Filtrando conteúdo com **grep**

Procurar um gene em um arquivo GTF:

```
grep "BRCA1" annotation.gtf
```

Filtrar todas as sequências que contenham "ATGC":

```
grep "ATGC" arquivo.fasta
```

Salvar resultado:

```
grep "ATGC" arquivo.fasta > filtradas.fasta
```



5. Extraíndo colunas com **cut**

Suponha um arquivo TSV com genes e expressão:

```
cut -f 1,3 arquivo.tsv
```

Significa: selecione coluna 1 e 3.

Para separar por vírgula:

```
cut -d ',' -f 2 arquivo.csv
```



6. Contando itens únicos com **sort** e **uniq**

Exemplo: Contar quantos identificadores distintos existem no FASTQ:

Cada read tem um identificador na **linha 1** do bloco de 4 linhas, começando com **@**.

```
grep "^@" arquivo.fastq | cut -d ' ' -f 1 | sort | uniq -c
```

O que faz:

- extrai headers (**@M01234...**)
- mantém somente a primeira parte do identificador
- ordena

- conta quantas vezes cada ID aparece

Contar quantas reads começam com o mesmo padrão (ex: 6 primeiras bases)

Útil para verificar over-representation ou artefatos técnicos:

```
awk 'NR % 4 == 2 {print substr($0,1,6)}' arquivo.fastq | sort | uniq -c
```

O que faz:

- pega somente as **linhas de sequência** (linha 2 de cada bloco de 4)
- extrai as primeiras 6 bases
- conta padrões distintos

Exemplos:

- ✓ Contar padrões nas sequências de um FASTQ

```
awk 'NR % 4 == 2 {print substr($0,1,6)}' arquivo.fastq | sort | uniq -c
```

- ✓ Contar bases iniciais

```
awk 'NR % 4 == 2 {print substr($0,1,1)}' arquivo.fastq | sort | uniq -c
```

- ✓ Contar sequências duplicadas

```
awk 'NR % 4 == 2 {print}' arquivo.fastq | sort | uniq -c
```

- ✓ Contar identificadores únicos

```
grep "^@" arquivo.fastq | cut -d ' ' -f 1 | sort | uniq -c
```

7. Pipes: encadeando comandos para análises rápidas

O pipe `|` envia a saída de um comando como entrada para outro.

Exemplo: contar quantas reads começam com "ATG":

```
grep "^ATG" arquivo.fastq | wc -l
```



8. Redirecionamento: criando arquivos de saída

Criar arquivo novo:

```
echo "resultado" > arquivo.txt
```

Acrescentar sem apagar o conteúdo:

```
echo "nova linha" >> arquivo.txt
```

Salvar filtragem:

```
grep "ATGC" arquivo.fastq > filtrado.fastq
```

Salvar relatório completo:

```
wc -l arquivo.fastq > resumo.txt
```

Exercícios

Exercício 1 — Contar reads

Use `wc -l` e `awk` para contar quantas reads existem em um FASTQ.

Exercício 2 — Filtrar sequências

Filtre todas as linhas que possuem "AAA" e salve em `aaa_reads.txt`.

Exercício 3 — Mini análise

Usando pipes, gere um relatório com:

- número total de reads
- número de reads contendo "ATG"
- % de reads com "ATG"