Análise de dados de Metagenômica

Dalmolin Systems Biology Group

Table of contents

Processamento e Análise de Dados de Metagenoma

Neste repositório está o material para o curso **Análise de dados de Metagenômica**, organizado pelo Prof. Rodrigo Dalmolin, do Centro Multiusuário de Bioinformática da UFRN.

O curso é dividido em 5 módulos:

- Controle de qualidade e pré-processamento
- Montagem
- Classificação taxonômica
- Anotação funcional
- Análises downstream:
  - Cálculos de diversidade e visualizações
  - Visualizações da anotação funcional

1 Organizando Ambiente para Análise

Primeiro, é necessário criar e ativar um ambiente Conda que contenha todas as ferramentas necessárias para os próximos passos. Caso ainda não tenha o Conda instalado, siga as instruções neste link.

```
$ conda env create -f medusaPipeline.yml
$ conda activate medusaPipeline
```

A estrutura de diretórios a seguir ajudará na organização dos arquivos baixados, arquivos intermediários e seus outputs:

```
$ mkdir -p ./Pipeline/{result,data/{merged,assembled,collapsed,removal/{index,reference},raw
```

2 Controle de Qualidade e Pré-processamento

⚠ WORK IN PROGRESS

2.1 Baixar Amostras

Mude para o diretório `Pipeline/data` e baixe os arquivos de exemplo em pares (pair-end):

```
$ cd Pipeline/data
$ fasterq-dump SRR579292 -e 8
```

Nota

O argumento `-e` no comando `fasterq-dump` especifica o número de threads que serão utilizadas. Adapte este argumento conforme o desempenho do seu sistema.

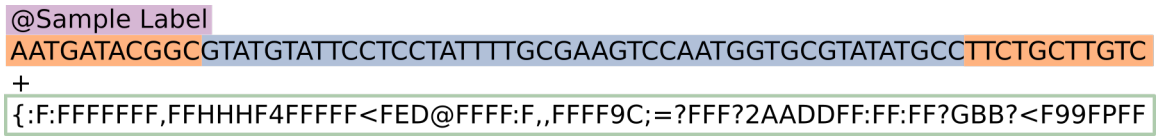
2.2 Controle de Qualidade com FastQC e MultiQC

2.2.1 Estrutura de Arquivos `.fastq`

O formato `.fastq` contém as sequências de DNA geradas no sequenciamento, junto com informações sobre a qualidade de cada nucleotídeo. A estrutura de um arquivo `.fastq` segue um padrão repetitivo a cada fragmento sequenciado:

1. **Linha 1:** Contém o identificador único da leitura, que pode incluir informações sobre o sequenciador e a amostra (começa com o símbolo `@`).
2. **Linha 2:** A sequência de nucleotídeos resultante do sequenciamento.
3. **Linha 3:** Opcionalmente usada para anotações ou descrições adicionais (começa com um símbolo `+`).
4. **Linha 4:** A qualidade de cada base na sequência, representada pelo Phred Score, que reflete a confiabilidade da leitura.

A imagem abaixo exemplifica essa estrutura:



- **Em Roxo:** Identificador do sequenciamento.
- **Em Laranja:** Sequências adaptadoras (caso ainda não tenham sido removidas), que podem variar dependendo da plataforma de sequenciamento.
- **Em Azul:** O fragmento de DNA sequenciado, conhecido como "*Insert Size*".
- **Em Verde:** Phred Score, que indica a qualidade de cada base na sequência.

2.2.2 Ferramentas de Controle de Qualidade

Para garantir que as sequências obtidas sejam de boa qualidade e adequadas para análise, utilizamos ferramentas como o **FastQC** e o **MultiQC**.

- **FastQC:** Avalia a qualidade de cada arquivo de sequenciamento individualmente, gerando relatórios com informações sobre a qualidade das bases, conteúdo GC, presença de adaptadores e outros aspectos que podem impactar a análise.
- **MultiQC:** Agrega os relatórios gerados pelo FastQC (ou outras ferramentas), criando um único relatório consolidado que facilita a visualização e interpretação dos dados de múltiplas amostras.

2.2.3 Gerando Relatórios de Qualidade

Para gerar relatórios de controle de qualidade com o FastQC para cada uma das amostras baixadas, use o seguinte comando:

```
# Gerando relatórios de qualidade para cada amostra
for sample in $(ls Pipeline/data/*.fastq.gz);
do
fastqc $sample -o Pipeline/data
done
```

Gere o relatório consolidado contendo todas as amostras a partir do output do FastQC:

```
# Consolidando relatórios com MultiQC
$ multiqc Pipeline/data
```

Interpretando os resultados do MultiQC

Verifique os gráficos de qualidade, presença de contaminantes e distribuições de qualidade das bases. Ajustes podem ser necessários para garantir a integridade dos dados para as etapas seguintes.

3 Montagem

3.1 Contexto

As *reads*, ou leituras, fragmentos de sequências gerados pelo processo de sequenciamento, podem ser re-organizadas e mescladas em sequências mais longas e contíguas, ou *contigs*. Esse processo é denominado de **Montagem**.

Para se realizar montagem, você pode utilizar uma referência, como o genoma referência de um organismo, que servirá como base para organizar as *contigs*. No entanto, no contexto metagenômico, a modalidade de montagem geralmente realizada é a montagem livre de referência, ou montagem *de novo*.

Como montador *de novo*, utilizaremos o **MEGAHIT**.

3.2 Realizando a montagem

Vamos montar as leituras pós-descontaminação usando o MEGAHIT:

```
megahit -1 ../removal/unaligned_1.fastq -2
../removal/unaligned_2.fastq -o SRR579292 -t 8
```

O arquivo de saída `SRR579292.contigs.fa` contém as sequências contíguas correspondentes às leituras usadas.

> **Nota**
>
> Os passos seguintes, como a classificação taxonômica, podem se utilizar tanto das leituras quanto dos contigs. Iremos utilizar as leituras por motivos de didática, mas a maioria das ferramentas utilizadas para a análise de metagenômica podem fazer uso tanto de *reads* quanto de *contigs*.

4 Classificação Taxonômica

4.1 Contexto

4.2 Realizando a classificação taxonômica

Mude o diretório atual para "Pipeline/taxonomic/db" e baixe os seguintes arquivos:

```
$ wget ftp://ftp.ncbi.nlm.nih.gov/pub/taxonomy/taxdump.tar.gz
$ wget ftp://ftp.ncbi.nlm.nih.gov/pub/taxonomy/accession2taxid/prot.accession2taxid.gz
```

Esses são os bancos de dados que utilizaremos para executar o Kaiju

Extraia e descompacte os arquivos necessários:

```
$ tar -xf taxdump.tar.gz nodes.dmp names.dmp
$ pigz -d prot.accession2taxid.gz -p 8
```

Construa um índice Kaiju:

```
$ kaiju-convertNR -t nodes.dmp -g prot.accession2taxid -e ~/miniconda3/envs/medusaPipeline/bi
$ kaiju-mkbwt -n 8 -a ACDEFGHIKLMNPQRSTVWY -o kaijuNR kaijuNR.fasta
$ kaiju-mkfmi kaijuNR
```

Nota: Na chamada `kaiju-convertNR`, por padrão, são incluídas apenas sequências de Archaea, Bactérias e Vírus do NCBI-nr. Este comportamento pode ser alterado com o argumento `-l`, passando um arquivo de entrada como `~/miniconda3/envs/medusaPipeline/bin/kaiju-taxonlistEuk.ts` Este argumento utiliza apenas sequências com ancestrais listados no arquivo.

Mude o diretório atual para "Pipeline/taxonomic" e execute a classificação taxonômica:

```
$ kaiju -t db/nodes.dmp -f db/kaijuNR.fmi -i ../data/removal/unaligned_1.fastq -j ../data/re
```

Adicione os nomes dos táxons ao output:

```
$ kaiju-addTaxonNames -t db/nodes.dmp -n db/names.dmp -r superkingdom,phylum,class,order,fami
```

Gere os gráficos Krona:

```
$ kaiju2krona -t db/nodes.dmp -n db/names.dmp -i ../result/SRR579292_kaiju.out -o ../result/S
$ ktImportText -o ../result/SRR579292_krona.html ../result/SRR579292_kaiju2krona.out
```

5 Anotação Funcional

⚠ WORK IN PROGRESS

5.1 annotate

Para nosso primeiro exemplo, vamos utilizar a ferramenta `annotate`, que irá transferir identificadores de um resultado de alinhamento para identificadores funcionais, nesse caso termos do gene ontology.

Mas, para fazer isso, primeiro precisamos alinhar nosso dado a uma referência!

5.1.1 Alinhamento

5.1.2 Executando o annotate

- Vamos primeiro filtrar as colunas do nosso arquivo de referência para as que contém identificadores do GenBank e do Gene Ontology:

```
awk -F "\\t" '{if((\$7!="") && (\$18!="")){print \$18"\\t"\$7}}' idmapping_selected.tab > ge
```

- E fazer o mesmo para termos um arquivo que traduz identificadores RefSeq para Gene Ontology:

```
awk -F "\\t" '{if((\$4!="") && (\$7!="")){print \$4"\\t"\$7}}' idmapping_selected.tab > refs
```

- Podemos então executar um script R que irá limpar os dois arquivos, preparando-os para o formato do annotate:

⚠ Aviso

O script requer bastante memória para lidar com o arquivo de mapeamento do UniProt, podendo requerir até 80GB de memória RAM para um dicionário típico.

```
createDictionary.R \
        NR2GO.txt \
        genbank2GO.txt \
        refseq2GO.txt \
        4
```

- Em sequência, criamos o banco de dados do annotate:

```
annotate createdb NR2GO.txt NR2GO 0 1 -d db
```

- E executamos o annotate para anotar cada *query* do alinhamento para seu identificador funcional:

```
annotate idmapping ../alignment/all_matches.m8 ../result/SRR579292_functional_GO.txt NR2GO -l
```

> **Nota**
>
> O argumento `-l` determina qual o comprimento mínimo de um alinhamento para ele ser considerado.

- Ao explorar o arquivo de resultado do annotate podemos ver que ele possui o seguinte formato: ...

6 Análises Downstream

> ⚠ WORK IN PROGRESS

6.1 Cálculos de Diversidade

6.2 Extraindo informação funcional