Análise de dados de Metagenômica

Dalmolin Systems Biology Group

Índice

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.

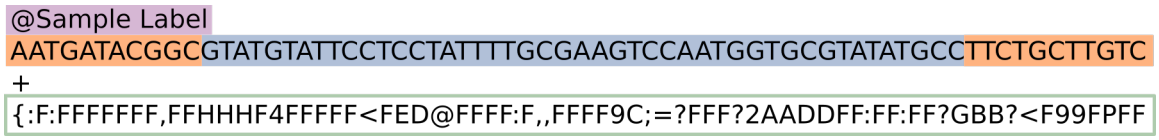
2.2 Controle de Qualidade com FastQC e MultiQC

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.

2.3 Pré-processamento das leituras

2.3.1 Contexto

Como vimos na seção anterior, nosso arquivo pode ter sequências de baixa qualidade, sejam estas sequências adaptadoras ou simplesmente erros no procedimento do sequenciamento. Portanto, devemos remover essas sequências de nossas análises posteriores, a fim de não permitir que interfiram com ruído na informação biológica real.

Para isso, obteremos uma amostra de sequenciamento (que será utilizada em todas as seções posteriores) e a processaremos com ferramentas de controle de qualidade, além de alinhar a amostra, de origem na microbiota humana, contra o genoma referência do hospedeiro, garantido que tenhamos apenas sequências advindas da microbiota.

2.3.2 Obtendo os dados

Mude o diretório atual para `"Protocol/data"` e baixe os dados de exemplo:

```
$ prefetch SRR579292 -X 20G
```

Nota

No comando `prefetch`, o argumento `-X` especifica o tamanho máximo do arquivo a ser baixado (adapte esse argumento se necessário).

Obtenha as leituras paired-end:

```
$ fasterq-dump SRR579292 -e 8
```

Nota

No comando `fasterq-dump`, o argumento `-e` especifica o número de threads a serem usadas (adapte esse argumento se necessário).

2.3.3 Removendo sequências de baixa qualidade

Remova sequências de baixa qualidade e adaptadores:

```
$ fastp -i SRR579292_1.fastq -I SRR579292_2.fastq -o trimmed/SRR579292_1_trim.fastq -O trimme
```

> **Nota**
>
> No comando `fastp`, o argumento `-q` define o limiar da pontuação de qualidade Phred, e `-w` especifica o número de threads a serem usadas (adapte esses argumentos se necessário).

Um genoma de referência de *Homo sapiens* é necessário para remover as sequências do hospedeiro desses dados. Mude o diretório atual para `"Pipeline/data/removal/reference"` e baixe um genoma de referência do Ensembl.

2.3.4 Remoção de Sequências do Hospedeiro

Baixe o genoma de referência de *Homo sapiens* do Ensembl:

```
$ wget ftp://ftp.ensembl.org/pub/release-102/fasta/homo_sapiens/dna/Homo_sapiens.GRCh38.dna.p
$ pigz -d Homo_sapiens.GRCh38.dna.primary_assembly.fa.gz -p 8
```

> **Nota**
>
> O software `pigz` é uma implementação paralela do `gzip`. O argumento `-p` especifica o número de threads a serem usadas.

Construa um índice Bowtie2:

```
$ bowtie2-build Homo_sapiens.GRCh38.dna.primary_assembly.fa ../index/host --threads 8
```

Volte para `"Pipeline/data"` e alinhe as sequências contra a referência:

```
$ bowtie2 -x removal/index/host -1 trimmed/SRR579292_1_trim.fastq -2 trimmed/SRR579292_2_trim
```

Extraia as leituras não alinhadas:

```
$ samtools view -bS removal/all.sam > removal/all.bam
$ samtools view -b -f 12 -F 256 removal/all.bam > removal/unaligned.bam
$ samtools sort -n removal/unaligned.bam -o removal/unaligned_sorted.bam
$ samtools bam2fq removal/unaligned_sorted.bam > removal/unaligned.fastq
$ cat removal/unaligned.fastq | grep '^@.*/1$' -A 3 --no-group-separator > removal/unaligned
$ cat removal/unaligned.fastq | grep '^@.*/2$' -A 3 --no-group-separator > removal/unaligned
```

> **Nota**
>
> No comando **SAMtools**, os argumentos **-f** e **-F** utilizam flags para especificar, respectivamente, quais alinhamentos devem ser extraídos e quais não devem ser extraídos. Consulte esta documentação para saber mais sobre as flags disponíveis no SAMtools.

Una as leituras paired-end:

```
$ fastp -i removal/unaligned_1.fastq -I removal/unaligned_2.fastq -o trimmed/unmerged_1.fasto
```

Remova sequências duplicadas:

```
$ fastx_collapser -i merged/SRR579292_merged.fastq -o collapsed/SRR579292.fasta
```

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

⚠ WORK IN PROGRESS

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
```

ℹ 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.tsv`. 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,fam
```

Gere os gráficos Krona:

```
$ kaiju2krona -t db/nodes.dmp -n db/names.dmp -i ../result/SRR579292_kaiju.out -o ../result/
$ 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

5.1.1.1 Adquirindo o banco de dados

Mude o diretório atual para `"Pipeline/alignment/db"` e baixe o banco de dados de proteínas do NCBI-nr:

⚠ Aviso

O banco de dados é **muito grande** e **inviável** de se instalar em um computador pessoal. Use um cluster de alta performance ou ambientes similares!

```
$ wget ftp://ftp.ncbi.nlm.nih.gov/blast/db/FASTA/nr.gz
$ pigz -d nr.gz -p 8
```

Construa um índice DIAMOND:

```
$ diamond makedb --in nr -d ../index/nr
```

5.1.1.2 Realizando o alinhamento

Mude o diretório atual para `"Pipeline/alignment"` e alinhe as leituras contra o banco de dados de proteínas de referência:

```
$ touch unaligned.fasta
$ diamond blastx -d index/nr -q ../data/collapsed/SRR579292.fasta -o matches.m8 --top 3 --un
```

> **Nota**
>
> No comando DIAMOND, o argumento `--un` especifica o arquivo usado para gravar as sequências não alinhadas, e `--top 3` reporta alinhamentos dentro dessa porcentagem da melhor pontuação de alinhamento.
> O DIAMOND pode usar muita memória e espaço temporário em disco. Portanto, o programa pode falhar ao esgotar um desses recursos. O argumento `-b` especifica o tamanho do bloco e `-c` o número de partes para processamento do índice. O uso total de memória pode ser estimado aproximadamente como **2(b+9b/c) GB**. Em um servidor com alta memória, defina `-c 1`.

Realize um alinhamento mais sensível usando as sequências não alinhadas:

```
$ touch unaligned2.fasta
$ diamond blastx -d index/nr -q unaligned.fasta -o matches2.m8 --more-sensitive --top 3 --un
```

> **Nota**
>
> O modo padrão sensível é projetado para leituras curtas (~100 pb), encontrando hits com >60% de identidade. Para sequências mais longas, os modos sensíveis são recomendados. O DIAMOND é muito mais eficiente para arquivos grandes (>1 milhão de leituras).

Verifique o número de consultas que apresentam hits com pelo menos 80% de identidade em `matches2.m8`:

```
$ awk '{ if ($3>=80) { print } }' matches2.m8 > check.m8
```

```
$ awk '{a[$1]++}END{for (i in a) sum+=1; print sum}' check.m8
```

5.1.1.3 Concatenar os resultados do alinhamento:

```
$ cat matches.m8 matches2.m8 > all_matches.m8
```

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 -
```

> **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:

```
Query    Annotation
342-2    GO:0005829; GO:0008720; GO:0047964; GO:0030267; GO:0016618; GO:0051287
1560-1   GO:0005829; GO:0005886; GO:0003854; GO:0102294; GO:0070403; GO:0016616; GO:0004769; G
3588-1   GO:0000775; GO:0005737; GO:0005829; GO:0090443; GO:0110085; GO:0044732; GO:0005634; G
5204-1   GO:0016491; GO:0008202
7527-1   Unknown
9319-1   GO:0042802; GO:0030170; GO:0008483; GO:0006520; GO:0009058
9922-1   GO:0000775; GO:0005737; GO:0005829; GO:0090443; GO:0110085; GO:0044732; GO:0005634; G
10306-1  Unknown
12743-1  GO:0005524; GO:0140658; GO:0004386; GO:0016787
```

Para cada busca ("Query") do seu alinhamento original, temos uma ou mais anotações em termos do Gene Ontology ("Annotation")

5.2 eggNOG-mapper

O eggNOG-mapper é uma ferramenta que realiza anotação funcional de sequências contíguas. Ele usa informação de ortologia e filogenia para transferir identificadores funcionais às sequências. O eggNOG-mapper pode ser executado através de uma interface web mas aqui ilustraremos seu uso na linha de comando.

Para isso, vamos utilizar as sequências contíguas que montamos anteriormente, na Seção 3.

- O eggNOG-mapper pode ser executado da seguinte maneira:

```
emapper.py \
        -m diamond \
        --itype metagenome \
        -i SRR579292.contigs.fa \
        -o eggnog_results/SRR579292 \
        --cpu 6
```

As opções que utilizamos foram:

- `-m`: Sinaliza o "modo" de execução do eggnog, neste caso a ferramenta que será utilizada para realizar a busca por sequências. No nosso caso utilizamos o DIAMOND, ferramenta que já conhecemos na Sessão 5.1.1.
- `--itype`: Sinaliza o tipo de entrada (*input*) que damos a ferramenta, no nosso caso sequência contíguas originadas de um metagenoma.
- `-i`: Determina a entrada para a ferramenta.
- `-o`: Determina para onde deve ser escrita a saída da ferramenta.
- `--cpu`: O número de núcleos de processamento a serem utilizados pela ferramenta.

Uma vez que executamos o eggNOG-mapper, teremos um resultado no seguinte formato (`eggnog_results/SRR579292.emapper.annotations`):

```
#query  seed_ortholog   evalue  score   eggNOG_OGs      max_annot_lvl   COG_category    Desc
k141_94572      552396.HMPREF0863_01053 4.43e-45        159.0   COG0595@1|root,COG0595@2|Bac
k141_47286      1236514.BAKL01000034_gene2896   3.35e-79        254.0   COG5009@1|root,COG50
k141_141858     158189.SpiBuddy_1606    8.08e-76        239.0   COG1593@1|root,COG1593@2|Bac
k141_23643      272563.CD630_25090      3.3e-45 159.0   COG1486@1|root,COG1486@2|Bacteria,1T
k141_70929      1122985.HMPREF1991_02897        3.54e-11        65.5    COG1595@1|root,COG15
```

Bastante informação! Mas essencialmente semelhante à ferramenta anterior: Temos em cada linha uma sequência de busca advinda do nosso arquivo de montagem, e em cada uma das outras colunas, anotações para diferentes bancos de dados de informação funcional, além de colunas detalhando a qualidade de alinhamnto das nossas sequências à esses identificadores.

Para mais informações sobre o arquivo de saída do eggNOG-mapper, confira a documentação da ferramenta.

6 Análises Downstream

⚠ WORK IN PROGRESS

6.1 Cálculos de Diversidade

6.2 Extraindo informação funcional