PIMBA v3.0 em Snakemake - Guia de Usuário

Autores: Tiago Ferreira Leão
Renato R. M. Oliveira
Versão do documento: 1.2
Data: 05/05/2025

O PIMBA (PIpeline for MetaBarcoding Analysis)(Oliveira et al. 2021) é uma ferramenta para análise de *metabarcoding* que permite ao usuário criar seu próprio banco de dados, superando limitações de outras ferramentas semelhantes. O PIMBA adapta o pipeline Qiime/BMP (Bolyen et al. 2019)(Pylro et al. 2014) para o clustering de OTUs e inclui correções opcionais de OTU usando o algoritmo LULU (Frøslev et al. 2017). Ele suporta leituras pareadas e não pareadas (com opções de índice único ou duplo), e permite a inferência de ASVs usando o Swarm (Mahé et al. 2021). O PIMBA fornece análises preliminares de abundância e diversidade automaticamente. Esse pipeline está escrito em Snakemake (Köster and Rahmann 2012), um sistema de gerenciamento de fluxo de trabalho projetado para agilizar a execução de pipelines complexos de análise de dados, oferecendo vantagens significativas em relação aos scripts em *bash* tradicionais. O Snakemake fornece uma abordagem estruturada e modular que melhora a legibilidade e a capacidade de gerenciamento do pipeline. Este guia irá ajudá-lo a instalar e rodar o PIMBA usando o Snakemake.

Pré-requisitos

Antes de instalar o software, certifique-se de ter o seguinte:

- Um sistema operacional baseado em Linux (por exemplo, Ubuntu, CentOS, Fedora) ou Windows WSL (*Windows Subsystem for Linux*);
- Python (versão 3.5 ou posterior) instalado em seu sistema;
- Git.

Instalação

A) Anaconda

Para rodar o Snakemake é necessário instalar o Anaconda seguindo os passos:

1. Baixar o arquivo de instalação aqui: https://www.anaconda.com/download/;
2. Dentro do terminal e na pasta onde se encontra o arquivo de instalação, rodar:

```
bash <nome-do-arquivo>-latest-Linux-x86_64.sh
```

Por exemplo:

```
bash Anaconda3-2024.10.1-latest-Linux-x86_64.sh
```

3. Siga as instruções nas telas do instalador. Se você não tiver certeza sobre qualquer configuração, aceite os padrões. Você pode alterá-los mais tarde;
4. Para que as alterações entrem em vigor, feche e reabra a janela do terminal;
5. Teste sua instalação. Na janela do terminal, execute o comando “conda list”. Uma lista de pacotes instalados aparece se tiver sido instalado corretamente.

B) Snakemake

Também é necessário criar um novo ambiente conda e instalar o Snakemake e o Singularity, de acordo com os seguintes passos:

```
conda create -n snakemake_env -c bioconda \
-c conda-forge singularity=3.8.6 snakemake=7.32.4

conda activate snakemake_env
```

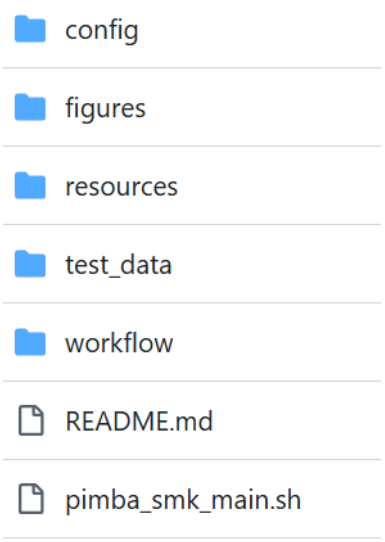
C) Clonar repositório do GitHub

Por fim, é necessário clonar o repositório do GitHub onde se encontra o código para rodar o PIMBA em Snakemake. Use o comando abaixo.

```
git clone https://github.com/itvgenomics/pimba_smk.git

cd pimba_smk/
```

Você deve observar os seguintes arquivos dentro do diretório clonado:



Esses arquivos consistem de:

- Config: diretório com o arquivo “config.yaml” onde todos os parâmetros de análise serão configurados. Para um usuário padrão, esse é o único arquivo que deve ser modificado (mais instruções de como modificá-lo virão a seguir);
- Resources: diretório com recursos necessários para rodar o PIMBA, como por exemplo o arquivo “adapters.txt” com sequência de adaptadores;
- Workflow: diretório com os scripts do Snakemake para rodar o PIMBA;
- Test_data: dados para realizar o teste do algoritmo;
- README.md: guia de usuário semelhante a este documento;
- pimba_smk_main.sh: script principal para rodar o PIMBA depois de editar corretamente o “config.yaml”.

Como rodar o PIMBA v3.0 em Snakemake

A) Configurar o arquivo “config.yaml”

O arquivo “config.yaml” (dentro da pasta “config”) é o arquivo geral de configurações. Nele deve conter os parâmetros como o número máximo de processadores, sequências para os adaptadores, caminhos para arquivos de input e etc. Abra o arquivo em um editor de texto e faça as seguintes modificações. **Atenção: usar caminhos completos, caminhos parciais não irão funcionar nesta versão.**

1. Número de processadores: A opção “num_threads” indica o número máximo de processadores que devem ser utilizados para tarefas que permitem paralelização;

```
# number of threads
num_threads: 8
```

2. Inputs gerais para rodar o modo prepare: caso o usuário deseje rodar o modo “prepare” para preparar as leituras que serão rodadas no modo “run”, editar as seguintes opções:

a. “minlength”: o tamanho mínimo de uma leitura após o tratamento de qualidade;
b. “minphred”: o valor PHRED mínimo para o tratamento de qualidade;
c. “outputprepare”: nome do arquivo de output a ser criado no formato FASTA.

```
# pimba prepare general
minlength: 100
minphred: 20
outputprepare: 'AllSamples'
```

3. Inputs para rodar leituras pareadas: caso o usuário deseje rodar o PIMBA para leituras pareadas, é necessário configurar a “rawdatadir” com o caminho para o diretório onde encontram-se as leituras e o “adapters” com caminho para o arquivo de adaptadores dentro do diretório “resources”. Adicionalmente, selecione o mínimo de overlap (“minoverlap”) e o mínimo de similaridade (“minsim”) para leituras serem concatenadas. Selecione também o concatenador (“merger”) entre o PEAR ou o OverlapPER (escreva o nome todo com letras minúsculas, o padrão é o PEAR). Veja um exemplo abaixo e substitua com os caminhos corretos na sua máquina.

```
# pimba prepare paired end
rawdatadir: '/mnt/c/Users/c0519/Desktop/ITV/pimba_dev/pimba_smk/test_data/rawdata_COI/'
adapters: '/mnt/c/Users/c0519/Desktop/ITV/pimba_dev/pimba_smk/resources/adapters.txt'
minoverlap: 10
minsim: 0.9
merger: 'pear'
```

4. Input para leituras não-pareadas e single index (index único): caso o usuário tenha leituras não-pareadas e de single index, é necessário configurar as seguintes opções. Veja um exemplo abaixo e substitua com os caminhos corretos na sua máquina.
    a. “raw_fastq”: arquivo FASTQ de input;
    b. “prefix”: nome para ser incluído como prefixo dos arquivos a serem gerados;
    c. “barcodes_5end_txt”: caminho para o códigos de barras utilizados como índice nas pontas 5' das leituras;
    d. “singleadapter”: sequência de adaptador encontrado nas leituras;
    e. “barcodes_5end_fasta” = caminho do arquivo fasta para o “barcodes_5end_txt”.

```
# pimba prepare single end and single index
raw_fastq: '/home/tfleao/Desktop/ITV/pimba_training/rawdata_single/Chip1-2.fastq'
prefix: 'Chip1-2'
barcodes_5end_txt: '/home/tfleao/Desktop/ITV/pimba_training/pimba/barcodes_Rtags.txt'
singleadapter: 'TCCACTAATCACAAAGANATNGGNAC'
barcodes_5end_fasta: '/home/tfleao/Desktop/ITV/pimba_training/pimba/barcodes_Rtags.fasta'
```

5. Input para leituras não-pareadas e dual index (index duplo): caso o usuário tenha leituras não-pareadas e de dual index, é necessário configurar as seguintes opções. Veja um exemplo abaixo e substitua com os caminhos corretos na sua máquina.
    a. "raw_fastq": arquivo FASTQ de input;
    b. "barcodes_3end_txt": caminho para o códigos de barras utilizados como índice nas pontas 3' das leituras;

c. "barcodes_3end_rev": caminho do reverso-complemento do “barcodes_3end_txt”;
d. "barcodes_3end_fasta": caminho do arquivo fasta para o “barcodes_3end_txt”;
e. "barcodes_5end_dir": caminho para o diretório com todos os barcodes.fasta e barcodes.txt usados nos 5' das leituras. Cada código de barras de 3' deve ter um arquivo fasta e txt com todos os códigos de barras de 5' associados;
f. "forward_adapter": sequência do primer direto;
g. "reverse_adapter": sequência do primer reverso;

```
# pimba prepare single end and dual index
raw_fastq: '/home/tfleao/Desktop/ITV/pimba_training/rawdata_single/Chip1-2.fastq'
barcodes_3end_txt: '/home/tfleao/Desktop/ITV/pimba_training/pimba/barcodes_Rtags.txt'
barcodes_3end_rev: '/home/tfleao/Desktop/ITV/pimba_training/pimba/barcodes_reverse_Rtags_revcom.txt'
barcodes_3end_fasta: '/home/tfleao/Desktop/ITV/pimba_training/pimba/barcodes_Rtags.fasta'
barcodes_5end_dir: '/home/tfleao/Desktop/ITV/pimba_training/pimba/barcode_for_to_each_R/'
forward_adapter: 'TCCACTAATCACAAAGANATNGGNAC'
reverse_adapter: 'AGAAAATCATAATNAANGCNTGNGC'
```

6. Input para o modo “run”: após configurar o modo “prepare” de acordo com o tipo de leitura utilizada, deve-se configurar os inputs para o modo “run” de acordo com os parâmetros abaixo. Veja um exemplo abaixo e substitua com os caminhos corretos na sua máquina.
    a. "outputrun": nome do output a ser gerado;
    b. "strategy": estratégia de análise a ser usada. Pode ser “otu” ou “asv”. Se “otu”, o PIMBA usa vsearch. Se “asv”, o swarm é usado;
    c. "otu_similarity": porcentagem de similaridade usada no agrupamento de OTU. O padrão é 0.97;
    d. "assign_similarity": porcentagem de similaridade usada na atribuição de taxonomia. O padrão é 0.9;
    e. "mincoverage": cobertura mínima para o alinhamento. O padrão é 0.9;
    f. "otu_length": comprimento mínimo para cortar as leituras. Se o valor for 0, nenhuma leitura será cortada;
    g. "hits_per_subject": se 1, escolha o melhor hit. Se > 1, escolha por maioria. O padrão é 1;
    h. "marker_gene": gene marcador e banco de dados da análise. Pode ser: 16S-SILVA, 16S-GREENGENES, 16S-RDP, 16S-NCBI, ITS-FUNGI-NCBI, ITS-FUNGI-UNITE, ITS-PLANTS-NCBI ou COI-NCBI. O caminho para cada banco de dados deve ser configurado a seguir. Para um banco de dados customizado, insira o caminho por extenso para o diretório do banco;
    i. "e_value": valor esperado (e-value) usado pelo BLAST. O padrão é 0.001;
    j. "lulu": se definido como 'yes', o PIMBA descartará OTUs ou ASVs errôneas com LULU. O padrão é não usar LULU (‘no’);
    k. "ITS": definir como ‘yes’ caso as leituras sejam de ITS;
    l. "remote": definir se o BLAST será feito no modo remoto (sem precisar baixar o banco de dados) ou no modo local;
    m. “db_type”: definir o banco de dados BLAST do NCBI, por exemplo, nt, core_nt e assim por diante.

```
# pimba run
outputrun: 'AllSamples_97clust90assign'
strategy: 'otu'
otu_similarity: 0.97
assign_similarity: 0.9
mincoverage: 0.9
otu_length: 200
hits_per_subject: 1
marker_gene: 'COI-BOLD'
e_value: 0.001
lulu: 'no'
ITS: 'no'
remote: 'yes'
db_type: 'nt'
```

7. Caminhos para os bancos de dados: dependendo do banco de dados utilizado, fornecer o caminho completo para os arquivos referente a este banco de dados. O Snakemake apenas usará o caminho indicado na opção “marker_gene” do item anterior, portanto, somente o gene marcador indicado precisa ser configurado. Veja um exemplo abaixo e substitua com os caminhos corretos na sua máquina.

```
# database files
SILVA_DB_16S: '/home/tfleao/Desktop/ITV/pimba/Silva_132_release/SILVA_132_QIIME_release/'
COI_BOLD_DB: '/home/tfleao/Desktop/ITV/pimba_training/pimba/COI_BOLD/'
GG_DB_16S: '/home/tfleao/Desktop/ITV/pimba_training/pimba/gg_13_8_otus/'
RDP_DB_16S: '/home/tfleao/Desktop/ITV/pimba_training/pimba/RDP/'
NCBI_DB_EXP: '/home/tfleao/Desktop/ITV/pimba_training/pimba/ALL_NCBI/blastdb'
ITS_UNITE_DB: '/home/tfleao/Desktop/ITV/pimba_training/pimba/sh_refs_qiime_ver8.2/'
```

"taxdump": para bancos de dados NCBI, certifique-se de baixar os seguintes arquivos para /your/path/to/taxdump/ e incluir o caminho no config.yaml, baixe usando seu terminal como abaixo;

```
wget ftp://ftp.ncbi.nlm.nih.gov/pub/taxonomy/new_taxdump/new_taxdump.tar.gz

tar -xzvf new_taxdump.tar.gz
```

8. PIMBA plot: esta parte é referente a geração de gráficos para os resultados processados. Para rodar o PIMBA plot, configure “metadata” com o caminho para o arquivo de metadados e configure “group_by” com o parâmetro do metadado para agrupar as amostras (use “False” para não agrupar amostras). Veja um exemplo abaixo e substitua com os caminhos corretos na sua máquina.

```
# pimba plot
metadata: '/home/tfleao/Desktop/ITV/pimba_training/pimba/mapping_fileCOI.csv'
group_by: 'FALSE'
```

B) Rodar o arquivo “pimba_smk_main.sh”

O arquivo “pimba_smk_main.sh” é o script bash principal que roda todos os passos do pipeline em Snakemake. Este arquivo leva como input os seguintes parâmetros:

1. “-p”: modo de preparação do PIMBA; escolha entre “paired_end”, “single_index”, “dual_index” ou “no";
2. “ -r”: modo de execução PIMBA; especifique o nome do gene marcador (e consequentemente banco de dados) a ser utilizado, escolhendo entre 16S-SILVA, 16S-GREENGENES, 16S-RDP, 16S-NCBI, ITS-FUNGI-NCBI, ITS-FUNGI-UNITE, ITS-PLANTS-NCBI ou COI-NCBI. Para banco de dados customizado, incluir o caminho do diretório onde o banco está armazenado ao invés do gene marcador. Para pular indique “no”;
3. “-g”: modo de plotagem PIMBA: escolha entre “yes” ou “no”;
4. “-t”: número de processadores;
5. “-c”: o caminho para o arquivo config.yaml.
6. “-d”: o caminho para a pasta de trabalho.

Rode o pipeline com o seguinte comando no terminal (dentro da pasta clonada do GitHub):

```
bash pimba_smk_main.sh -p <modo de preparação do PIMBA> -r <modo de execução do PIMBA> \
-g <modo de plotagem do PIMBA> -t <número de processadores> -c <arquivo config.yaml> -d <pasta de trabalho>
```

Exemplo de teste: uso os dados na pasta “test_data” para testar o algoritmo rodando. Primeiro modifique os caminhos corretos no config (incluindo o caminho para o banco de dados BOLD) e depois rode o seguinte comando:

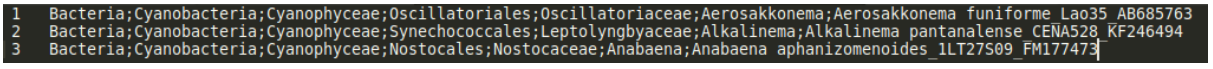
```
bash pimba_smk_main.sh -p paired_end -r COI-BOLD -g yes -t 8
-c config/config.yaml -d .
```

Observação: para rodar no cluster, existe um exemplo de .pbs no repositório.

Configurar banco de dados personalizado

Suponha que você queira usar um banco de dados personalizado. Nesse caso, você precisará apenas de um arquivo FASTA com as sequências de referência e suas identificações, além de um arquivo tax.txt de duas colunas contendo o ID da sequência e a taxonomia completa escrita para cada sequência de referência no arquivo FASTA. Coloque-os no mesmo diretório, por exemplo: "caminho/para/seu/database/".
Exemplo de arquivo FASTA:

Exemplo de arquivo de taxonomia:



Em seguida, instale o blastn em seu computador e execute makeblastdb no seu arquivo FASTA:

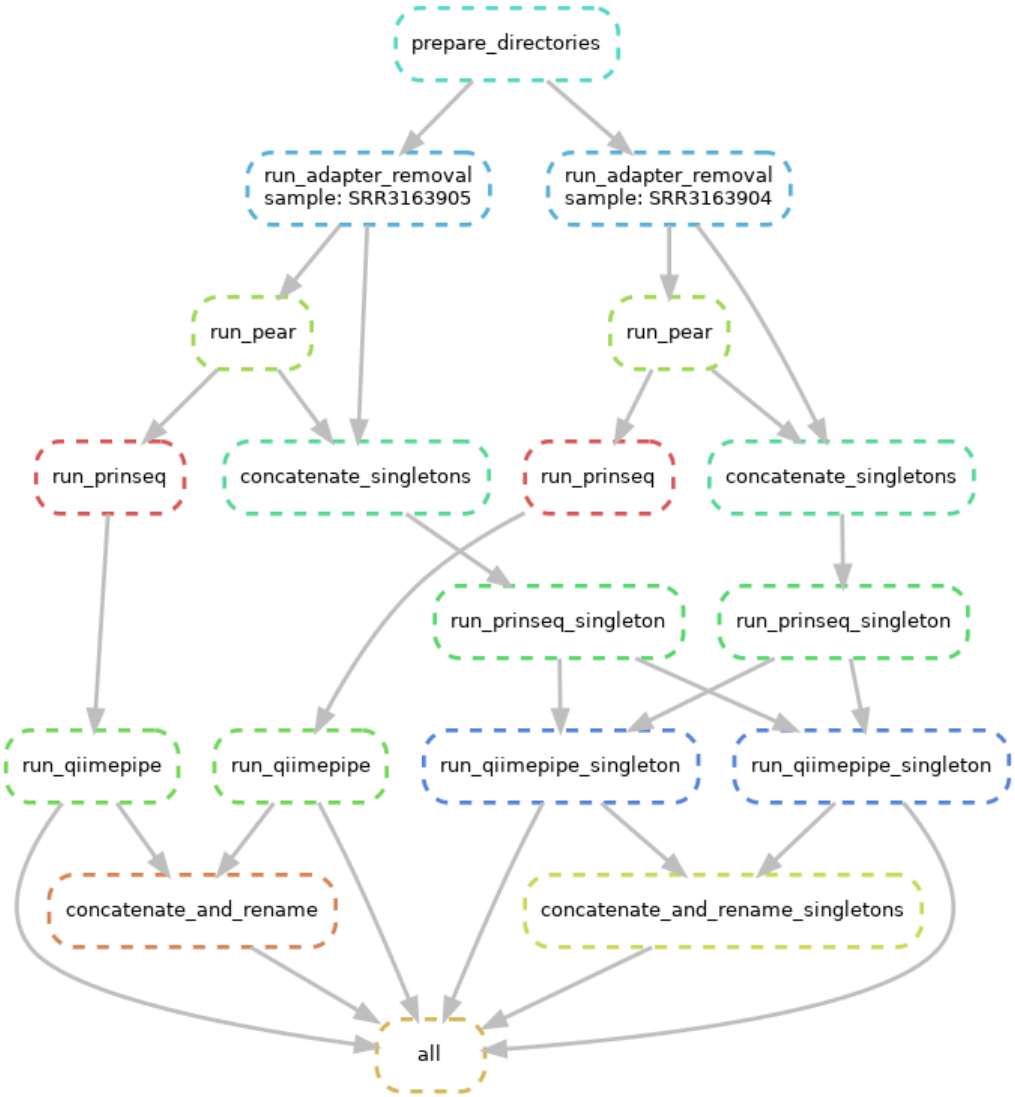
```
conda install bioconda::blast
makeblastdb -in <seu_arquivo_fasta.fasta> -dbtype nucl -parse_seqids
```

Depois disso, você precisa definir "caminho/para/seu/database/" na variável de configuração marker_gene do arquivo config.yaml e executar o script bash como neste exemplo:

```
bash pimba_smk_main.sh -p paired_end -r caminho/para/seu/database/ -g yes
-t 8 -c config/config.yaml
```

Passos envolvidos no PIMBA Snakemake pipeline

O Snakemake permite gerar DAGs (Gráficos Diretos Acíclicos, em inglês, Directed Acyclic Graphs) que indicam quais regras (“rules”) serão executadas pelo pipeline, ou seja, quais passos estão envolvidos na análise. Por exemplo, o DAG de uma de duas amostras de leituras de 16S rRNA pode ser visualizado abaixo. Esse DAG corresponde ao modo de preparação do PIMBA (“PIMBA prepare”) e percebe-se que vários passos ocorrem em paralelo, permitindo maior agilidade em processar as amostras se comparado ao script em bash.



Após processar o passo de preparação para essas duas amostras, o PIMBA Snakemake processa o modo de execução (modo “run”), ilustrado pelo segundo DAG abaixo. Neste DAG existem passos iniciais que não podem ser executados em paralelo, mas os passos finais (após a regra “run_bmp_renamer”) são executados com certa paralelização. Em ambos DAGs mostrados, a regra final chamada “all” é a que indica quais outputs estão faltando e quais respectivas regras devem ser rodadas para gerar esses outputs. Essa característica da regra “all” confere modularidade ao Snakemake e permite

que somente os outputs que faltam sejam gerados, o que ajuda em reiniciar o pipeline do ponto que parou quando há interrupções (diferentemente do bash que reinicia do começo).

Referências

Bolyen, Evan, Jai Ram Rideout, Matthew R. Dillon, Nicholas A. Bokulich, Christian C. Abnet, Gabriel A. Al-Ghalith, Harriet Alexander, et al. 2019. “Reproducible, Interactive, Scalable and Extensible Microbiome Data Science Using QIIME 2.” *Nature Biotechnology* 37 (8): 852–57.

Frøslev, Tobias Guldberg, Rasmus Kjøller, Hans Henrik Bruun, Rasmus Ejrnæs, Ane Kirstine Brunbjerg, Carlotta Pietroni, and Anders Johannes Hansen. 2017. “Algorithm for Post-Clustering Curation of DNA Amplicon Data Yields Reliable Biodiversity Estimates.” *Nature Communications* 8 (1): 1188.

Köster, Johannes, and Sven Rahmann. 2012. “Snakemake--a Scalable Bioinformatics Workflow Engine.” *Bioinformatics (Oxford, England)* 28 (19): 2520–22.

Mahé, Frédéric, Lucas Czech, Alexandros Stamatakis, Christopher Quince, Colomban de Vargas, Micah Dunthorn, and Torbjørn Rognes. 2021. “Swarm v3: Towards Tera-Scale Amplicon Clustering.” *Bioinformatics (Oxford, England)* 38 (1): 267–69.

Oliveira, Renato R. M., Raíssa Silva, Gisele L. Nunes, and Guilherme Oliveira. 2021. “PIMBA: A PIpeline for MetaBarcoding Analysis.” *Advances in Bioinformatics and Computational Biology*, 106–16.

Pylro, Victor S., Luiz Fernando W. Roesch, Daniel K. Morais, Ian M. Clark, Penny R. Hirsch, and Marcos R. Tótola. 2014. “Data Analysis for 16S Microbial Profiling from Different Benchtop Sequencing Platforms.” *Journal of Microbiological Methods* 107 (December):30–37.