Data mining in biosciences
M2 DMBS
2024-2025
Marie-Joe Karam Kassar

Open Reading Frame (ORF) detection in a partial human transcriptome assembly

1. Objective

In this project, you will extract Open Reading Frames (ORFs) from a partial human transcriptome assembly provided in the `Homo_sapiens_cdna_assembled.fasta` file. You will set up a Docker container to conduct the analysis in a controlled environment.

2. Prerequisites

Before proceeding, ensure you have:

- Docker installed on your system
- Basic knowledge of Python, Bash scripting, and Git
- Understanding of ORFs and transcriptome assembly

3. Project steps

Extract ORFs in Python

- Read the input FASTA file (`Homo_sapiens_cdna_assembled.fasta`).
- Detect ORFs based on start (ATG) and stop codons (TAA, TAG, TGA).
- Ignore overlapping or nested ORFs.
- Store results in a Python dictionary with the following structure:

```
{ORF_ID <str>: (sequence <str>, length <int>, source_contig <str>, (start
<int>, end <int>), reading_frame <int>)
}
```

Save Results

- Output ORFs in GFF format.
- Save ORF sequences in a multifasta file.

Validate ORFs with BLAST+

- Construct a BLAST database from the `uniprot_sprot.fasta` file as follow:

```
#! /bin/bash
makeblastdb -in /path/to/uniprot_sprot.fasta -dbtype prot -out
/path/to/uniprot_sprot
```

- Compare the obtained ORFs against `uniprot/swissprot` to identify coding sequences (CDS) and UTR regions by runnong BLAST. You can run BLAST from Python using the `subprocess` package, otherwise from bash.

```
#! /bin/bash
# 9606 is the taxid for human
blast? -db /path/to/uniprot_sprot -query /path/to/multifastafile -taxids 9606 -
outfmt 7 -out /path/to/output/file.tsv
```

- Parse the BLAST output file in order to validate the ORFs and to delimitate CDS and UTRs.
- Append your GFF file with the newly discovered features.
- Assess the false positive rate.

Plot ORF Size Distribution

- Generate a histogram of ORF lengths.
- Plot size distribution for true and false positives.

Estimating the Mean Length of Open Reading Frames (ORFs) Using a Geometric Distribution

The length of an ORF can be modeled using a geometric distribution, where each codon represents a trial, and encountering a stop codon is considered a "success". Assuming that each of the 64 possible codons in the genetic code is equally likely, p is the probability of encountering a stop codon in a single trial. Using the geometric distribution, determine the expected number of codons (trials) until a stop codon is encountered. Recall that for a geometric distribution, the expected number of trials until the first success is ☑ = $1/p$.

3. Project Setup

Your analysis must be run within a docker container.

Build a Docker Image

- Use `python:3.12-slim` as the base image.
- Install `blast+` (https://ftp.ncbi.nlm.nih.gov/blast/executables/blast+/LATEST/ncbi-blast-2.16.0+-x64-linux.tar.gz).
- Download the latest `uniprot/swissprot` database for human proteins.

`Dockerfile`:

```
FROM python:3.12-slim

RUN apt-get update && \
    apt-get install -y wget

WORKDIR /app
```

```
RUN wget https://ftp.ncbi.nlm.nih.gov/blast/executables/blast+/LATEST/ncbi-blast-
2.16.0+-x64-linux.tar.gz && \
    tar -xzvf ncbi-blast-2.16.0+-x64-linux.tar.gz && \
    mv ncbi-blast-2.16.0+ /usr/local/ && \
    rm ncbi-blast-2.16.0+-x64-linux.tar.gz

ENV PATH="/usr/local/ncbi-blast-2.16.0+/bin:$PATH"

RUN wget
https://ftp.uniprot.org/pub/databases/uniprot/current_release/knowledgebase/complet
e/uniprot_sprot.fasta.gz && \
    wget
https://ftp.uniprot.org/pub/databases/uniprot/current_release/knowledgebase/complet
e/reldate.txt && \
    gunzip uniprot_sprot.fasta.gz && \
    mkdir -p /db/uniprot && mv uniprot_sprot.fasta /db/uniprot && mv reldate.txt
/db/uniprot

COPY . /app
```

To build your Docker image and test it by running a container interactively, follow these steps:

- create an empty folder `dockerfiles/blast/v2.16.0`
- create a file called `dockerfile` within that folder and paster the above dockerfile content.
- from the `docker/images/blast/v2.16.0` folder run:

```
docker build -t blast:v2.16.0 .
```

Then go to your working directory containing your scirpt and input file and run a container as follows:

```
docker run -it --rm -v /path/to/your/workingdire:/analysis blast:v2.16.0 bash
```

4. Submission

Your final submission should be submitted on github with at least these files:

- Your `orf_detection.py` script.
- Your dockerfile.
- Output files: ORFs in GFF and multifasta formats.
- A README.md file explaining your project goal, results (plots and analysis results) and a discussion.

Version Control with Git

- Use explicit, unitary commits with meaningful messages.
- Submit the project on GitHub.