Reproducibility Runbook & Docker Setup

This document provides the necessary components to reproduce the analysis environment and workflow for the AI-Enhanced eDNA Pipeline.

1. Dockerfile

This Dockerfile creates a container with all necessary dependencies, including Conda, R, Python, and the specific libraries used in the pipeline.

```
# Use a base image with Conda pre-installed
FROM mambaorg/micromamba:1.4.1

# Set the working directory
WORKDIR /pipeline

# Copy the environment file into the container
COPY environment.yml .

# Create and activate the conda environment from the file
RUN micromamba create -n edna-ai-env -f environment.yml && \
    micromamba clean --all --yes

# Add shell command to automatically activate the environment upon login
SHELL ["micromamba", "run", "-n", "edna-ai-env", "/bin/bash", "-c"]

# The container is now ready. The user will mount their data and scripts.
ENTRYPOINT ["/bin/bash"]
```

environment.yml file:

```
name: edna-ai-env
channels:
  - conda-forge
  - bioconda
  - pytorch
  - nvidia
dependencies:
  - python=3.10
  - r-base=4.2
  - cutadapt
  - r-dada2
```

- blast
- pytorch
- torchvision
- torchaudio
- pytorch-cuda=11.8
- numpy
- pandas
- matplotlib
- scikit-learn
- hdbscan
- biopython
- psycopg2-binary
- streamlit

2. How to Build and Run

1. **Place Files:** Place the Dockerfile and environment.yml in the root of your project directory.
2. **Build the Docker Image:**
   docker build -t edna-ai-pipeline .

3. **Run the Container:** This command launches the container and mounts your project's preprocessing directory into the container's /pipeline directory.
   # Make sure you are in your project's root directory
   docker run -it --gpus all \
     -v $(pwd)/preprocessing:/pipeline/preprocessing \
     -p 8501:8501 \
     --name edna-container edna-ai-pipeline

   - --gpus all: Provides the container with access to your NVIDIA GPU.
   - -v: Mounts your local code and data into the container.
   - -p 8501:8501: Exposes the port for the Streamlit dashboard.

3. Full Pipeline Execution Script

This run_full_pipeline.sh script, placed inside your scripts directory, automates the entire workflow.

#!/bin/bash

Exit immediately if a command exits with a non-zero status.
set -e

```
echo "--- STARTING AI-ENHANCED eDNA PIPELINE ---"

# --- STEP 1: PREPARE BLAST DATABASES ---
echo "[INFO] Formatting FASTA databases for BLAST..."
python format_fasta_for_blast.py # Run this twice, once for PR2 and once for MIDORI2

echo "[INFO] Creating BLAST database for 18S (PR2)..."
makeblastdb -in ../databases/pr2_blast_formatted.fasta -dbtype nucl -parse_seqids -out
../databases/pr2_blast_db

echo "[INFO] Creating BLAST database for COI (MIDORI2)..."
makeblastdb -in ../databases/midori2_co1_blast_formatted.fasta -dbtype nucl -parse_seqids
-out ../databases/midori2_co1_blast_db

# --- STEP 2: RUN AI CLUSTERING ---
echo "[INFO] Running AI Clustering for 18S marker..."
# The script should be edited to set MARKER = '18S'
python run_ai_clustering.py

echo "[INFO] Running AI Clustering for COI marker..."
# The script should be edited to set MARKER = 'COI'
python run_ai_clustering.py

# --- STEP 3: RUN HYBRID ANNOTATION ---
echo "[INFO] Running Hybrid Annotation for 18S marker..."
# The script should be edited to set MARKER = '18S'
python run_hybrid_annotation.py

echo "[INFO] Running Hybrid Annotation for COI marker..."
# The script should be edited to set MARKER = 'COI'
python run_hybrid_annotation.py

# --- STEP 4: POPULATE DATABASE ---
echo "[INFO] Populating cloud PostgreSQL database..."
# Ensure DATABASE_URL is set as an environment variable
python populate_database.py

# --- STEP 5: LAUNCH DASHBOARD ---
echo "[INFO] Launching Streamlit Dashboard..."
streamlit run app.py

echo "--- PIPELINE COMPLETE ---"
```