# Protein Sequence Alignment Pipeline

Overall Goal:

Create a program that takes DNA sequences, finds the protein-coding parts, aligns them as proteins, and then converts the protein alignment back into a DNA alignment that respects codon boundaries (so gaps are always 3 nucleotides long).

# Step-by-Step Implementation Instructions

Step 1: Load the Input Data

What to do:

Read a FASTA file containing DNA sequences

How to do it:

Use Python's file reading functions or a bioinformatics library like Biopython

For each sequence in the file, store:

The sequence ID/name

The actual DNA sequence as a string

Keep all sequences in a list or dictionary for easy access

Step 2: Find Protein-Coding Regions (ORFs)

What to do:

Scan each DNA sequence to find regions that could code for proteins

How to do it:

For each DNA sequence:

Look through all three possible reading frames (starting at position 0, 1, and 2)

Search for "ATG" (start codon)

When you find "ATG", continue reading three nucleotides at a time until you hit a stop codon ("TAA", "TAG", or "TGA")

If the region between start and stop is long enough (user-defined minimum length), save it as an ORF

Record which sequence it came from, start/end positions, which reading frame, and the DNA sequence

Step 3: Convert DNA to Protein Sequences

What to do:

Translate each ORF from DNA to protein

How to do it:

For each ORF found in Step 2:

Break the DNA sequence into groups of three nucleotides (codons)

Use a translation table to convert each codon to its corresponding amino acid

Store both the protein sequence and the original codons (you'll need the codons later)

Step 4: Organize Sequences by Similarity

What to do:

Figure out which protein sequences are most similar to each other

How to do it:

Compare all protein sequences to each other using k-mer similarity (compare short chunks)

Create a similarity score between every pair of sequences

Arrange the sequences in order from most similar to least similar (this helps with alignment quality)

Step 5: Align the Protein Sequences

What to do:

Create a multiple sequence alignment of the proteins

How to do it:

Start with the two most similar sequences from Step 4

Align them using Needleman-Wunsch (global alignment algorithm)

Then add the next most similar sequence to the growing alignment

Continue until all sequences are included in one big alignment

Output:

aligned protein sequences with gaps inserted where needed

Step 6: Convert Protein Alignment Back to DNA

What to do:

Create a DNA alignment that matches the protein alignment

How to do it:

For each position in the protein alignment:

If there's an amino acid, look up the original three-nucleotide codon that coded for it

If there's a gap in the protein alignment, insert "---" (three dashes) in the DNA alignment

This ensures that DNA gaps always respect codon boundaries

Step 7: Analyze Codon Positions

What to do:

Study how different positions within codons vary

How to do it:

For each codon position in the DNA alignment:

Separate the 1st, 2nd, and 3rd nucleotide positions

Calculate how variable each position is across sequences

Save these statistics (like percent identity) to a spreadsheet file

Step 8: Create Visualizations

What to do:

Make plots showing the variability patterns

How to do it:

Use matplotlib or seaborn to create graphs

Plot the variability at each codon position along the sequence

Save the plots as image files (PNG or PDF)

Step 9: Save All Results

What to do:

Write all outputs to files

How to do it:

Save the aligned protein sequences as a FASTA file

Save the codon-aware DNA alignment as a FASTA file

Save the statistics from Step 7 as a CSV file

Save the plots from Step 8 as image files

Put all output files in a specified results folder

Step 10: Create User Interface

What to do:

Make the program easy to run from command line

How to do it:

Use argparse to handle command-line arguments

Allow users to specify:

Input FASTA file

Output folder

Minimum ORF length

k-mer size for similarity comparison

The program should run the entire pipeline with these user settings

Step 11: Testing and Documentation

What to do:

Ensure the program works correctly and is well-documented

How to do it:

Write small tests for each function (ORF finding, translation, alignment, etc.)

Add clear docstrings to every function explaining what it does

Create a README file with:

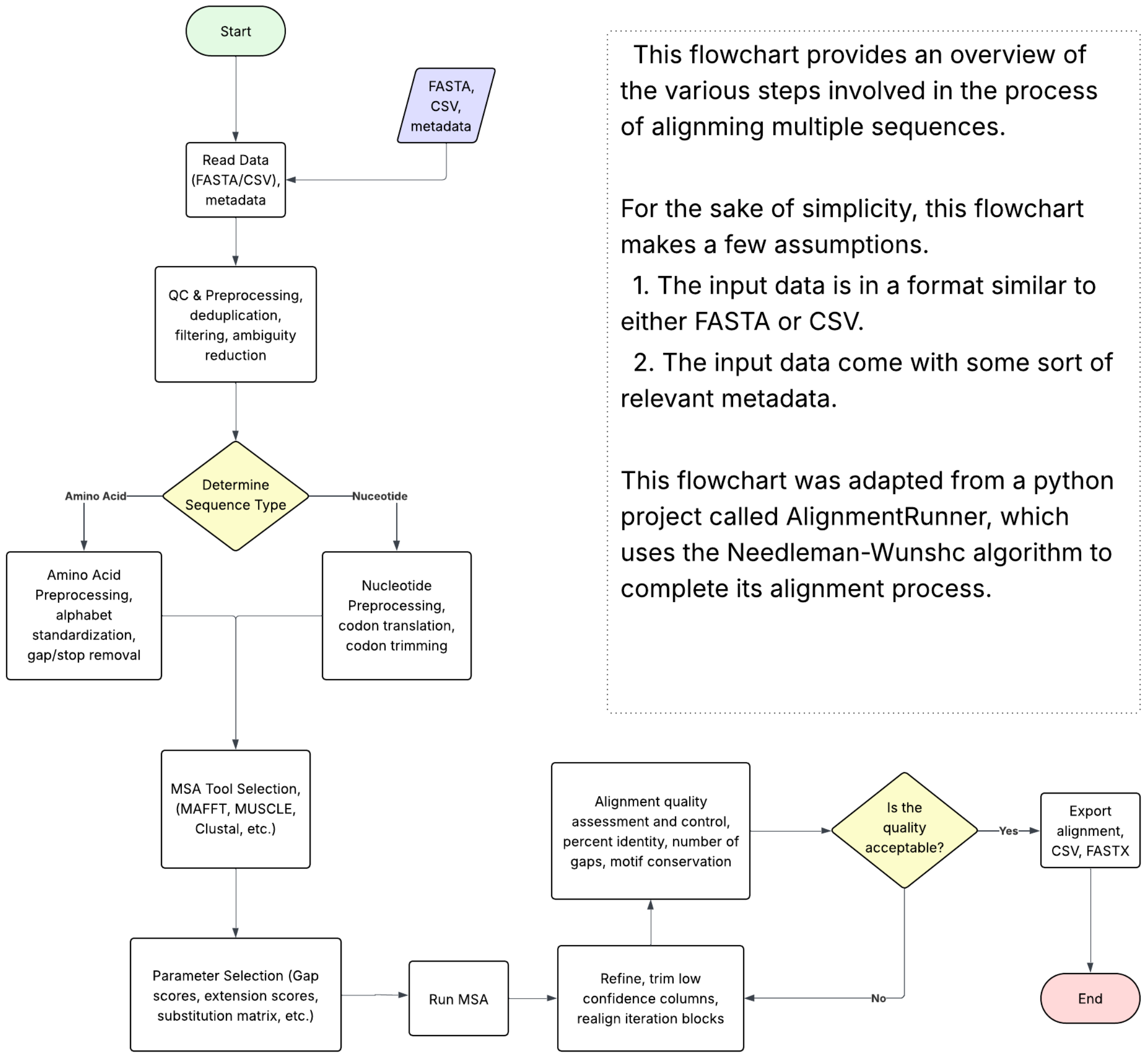
Installation instructions

How to run the program

Example commands

Description of what the program does

Note about any AI assistance used in development


Start
FASTA, CSV, metadata
Read Data (FASTA/CSV), metadata
QC & Preprocessing, deduplication, filtering, ambiguity reduction
Determine Sequence Type
Amino Acid
Nuceotide
Amino Acid Preprocessing, alphabet standardization, gap/stop removal
Nucleotide Preprocessing, codon translation, codon trimming
MSA Tool Selection, (MAFFT, MUSCLE, Clustal, etc.)
Parameter Selection (Gap scores, extension scores, substitution matrix, etc.)
Run MSA
Refine, trim low confidence columns, realign iteration blocks
Alignment quality assessment and control, percent identity, number of gaps, motif conservation
Is the quality acceptable?
Yes
No
Export alignment, CSV, FASTX
End

This flowchart provides an overview of the various steps involved in the process of alignming multiple sequences.

For the sake of simplicity, this flowchart makes a few assumptions.

1. The input data is in a format similar to either FASTA or CSV.
2. The input data come with some sort of relevant metadata.

This flowchart was adapted from a python project called AlignmentRunner, which uses the Needleman-Wunshc algorithm to complete its alignment process.