Translation-Based Sequence Alignment: Pseudocode and Flowchart

Expanded Pseudocode

CLI: msa_translate_align.py
parse_args() -> input_fasta, outdir, min_orf_len, genetic_code, kmer_k, seed, threads
set_random_seed(seed); create_outdirs(outdir/{aa,nt,stats,logs})
records = load_fasta(input_fasta) # id -> nt sequence
Preprocessing:
for id, nt in records:
nt = canonicalize(nt) # uppercase, strip gaps, replace invalid with N
drop if length < min_orf_len * 3 after cleaning
ORF detection:
for id, nt in records:
frames = scan_six_frames(nt)
candidates = find_orfs(frames, min_len=min_orf_len, code=genetic_code)
score each candidate (length primary; tie-break coding score/KA/Ks-free heuristic)
best_orf = select_best(candidates); if none: log and continue
Translate to AA:
aa_seq = translate(best_orf, code=genetic_code, unknown=X, stop=* or trim)
aa_records[id] = aa_seq
Guide tree via k-mers:
profiles = kmer_profile(aa_records, k=kmer_k, normalize=true)
D = jaccard_distance_matrix(profiles)
tree = build_tree(D, method=UPGMA_or_NJ)
Progressive AA alignment:
pairwise_aligner = needleman_wunsch(subst=BLOSUM62, gap_open=10, gap_extend=1)
aa_msa = progressive_align(aa_records, tree, pairwise_aligner, profile_profile=SP score)
Back-translate to NT:
nt_msa = codon_map_back(aa_msa, best_orf_map, gap_symbol=---)
QC validation:
assert all_equal_length(nt_msa)
assert in_frame(nt_msa) and no_internal_stops(nt_msa, code)
gap_rate_ok = per_column_gap_fraction(nt_msa) < threshold
Outputs:
write_fasta(aa_msa, outdir/aa/aligned_aa.fasta)
write_fasta(nt_msa, outdir/nt/aligned_nt.fasta)
Stats & logging:
stats = codon_position_stats(nt_msa) # GC1/2/3, gap profile, entropy
save_csv(stats, outdir/stats/codon_stats.csv); plot_pngs(stats, outdir/stats/)
log_run(outdir/logs/run.json, args, counts, versions)

Pipeline Flowchart