151554_Augustyniak_Alicja_lab5

June 4, 2025

[5]:
```python
import os
import sys
import math
from scipy import linalg
import numpy as np

PDB1 = "1.pdb"
PDB2 = "0.pdb"

def minimal_angular_difference(deg1, deg2):
    raw = abs(deg1 - deg2) % 360
    return min(raw, 360 - raw)

def load_pdb(path):
    structure = {}
    with open(path, 'r') as file:
        for line in file:
            if line.startswith('ATOM'):
                res_num = int(line[22:26].strip())
                atom_name = line[12:16].strip()
                coords = np.array([
                    float(line[30:38]),
                    float(line[38:46]),
                    float(line[46:54])
                ])

                if res_num not in structure:
                    structure[res_num] = {}
                structure[res_num][atom_name] = coords
    return structure

def get_residues_with_P(structure):
    return [res_num for res_num in structure if 'P' in structure[res_num]]

def calculate_dihedral(p1, p2, p3, p4):
    b1 = p2 - p1
    b2 = p3 - p2
    b3 = p4 - p3

    b2_norm = b2 / np.linalg.norm(b2)

    v = b1 - np.dot(b1, b2_norm) * b2_norm
    w = b3 - np.dot(b3, b2_norm) * b2_norm

    x = np.dot(v, w)
    y = np.dot(np.cross(b2_norm, v), w)

    return math.degrees(math.atan2(y, x))

def collect_torsion_differences(struct1, struct2):
    torsions = {
        'beta':  ['P', "O5'", "C5'", "C4'"],
        'gamma': ["O5'", "C5'", "C4'", "C3'"],
        'delta': ["C5'", "C4'", "C3'", "O3'"]
    }

    deltas = []
    residues = get_residues_with_P(struct1)

    for res_num in residues:
        if res_num not in struct2:
            continue

        for angle_type, atoms in torsions.items():
            if all(atom in struct1[res_num] for atom in atoms) and \
               all(atom in struct2[res_num] for atom in atoms):

                coords1 = [struct1[res_num][atom] for atom in atoms]
                coords2 = [struct2[res_num][atom] for atom in atoms]

                angle1 = calculate_dihedral(*coords1)
                angle2 = calculate_dihedral(*coords2)

                delta = minimal_angular_difference(angle1, angle2)
                deltas.append(delta)

    return deltas

def compute_mcq(deltas_deg):

    if not deltas_deg:
        print("No torsion angles calculated - CQ undefined.")
        sys.exit(1)
```

```python
    deltas_rad = np.radians(deltas_deg)
    sum_sin = np.sum(np.sin(deltas_rad))
    sum_cos = np.sum(np.cos(deltas_rad))

    mcq_rad = math.atan2(sum_sin, sum_cos)
    return abs(math.degrees(mcq_rad))

def superimpose(A, B):

    centroid_A = np.mean(A, axis=0)
    centroid_B = np.mean(B, axis=0)
    A_centered = A - centroid_A
    B_centered = B - centroid_B

    cov_matrix = np.dot(A_centered.T, B_centered)

    U, S, Vt = linalg.svd(cov_matrix)
    V = Vt.T

    sign = np.sign(np.linalg.det(cov_matrix))
    S_matrix = np.array([
        [1, 0, 0],
        [0, 1, 0],
        [0, 0, sign]
    ])
    R = np.dot(V, np.dot(S_matrix, U.T))

    B2 = np.dot(B_centered, R)

    rmsd = np.sqrt(np.mean(np.sum((A_centered - B2)**2, axis=1)))

    B_aligned = B2 + centroid_A

    return rmsd, B_aligned

def compute_rmsd(struct1, struct2):

    residues1 = set(get_residues_with_P(struct1))
    residues2 = set(get_residues_with_P(struct2))
    common_residues = sorted(residues1 & residues2)

    if not common_residues:
        print("No common residues with P atoms")
        return float('nan')

    coords1 = np.array([struct1[res]['P'] for res in common_residues])
    coords2 = np.array([struct2[res]['P'] for res in common_residues])
    if len(coords1) >= 3:
        rmsd, _ = superimpose(coords1, coords2)
    else:
        rmsd = np.sqrt(np.mean(np.sum((coords1 - coords2)**2, axis=1)))

    return rmsd

def main():

#if len(sys.argv) != 3:
    #print(f"Usage: {os.path.basename(sys.argv[0])} <model1.pdb> <model2.pdb>")
    #sys.exit(1)

#file1 = sys.argv[1]
#file2 = sys.argv[2]

 #if not os.path.isfile(file1) or not os.path.isfile(file2):
    #print(f"File '{file1}' or '{file2}' not found.")
    #sys.exit(1)

    if not os.path.isfile(PDB1) or not os.path.isfile(PDB2):
        print(f"File '{PDB1}' or '{PDB2}' not found.")
        sys.exit(1)

    struct1 = load_pdb(PDB1)
    struct2 = load_pdb(PDB2)

    residues1 = get_residues_with_P(struct1)
    residues2 = get_residues_with_P(struct2)

    if len(residues1) != len(residues2):
        print("Different number of residues with P atoms - exiting.")
        sys.exit(1)

    rmsd_val = compute_rmsd(struct1, struct2)
    print(f"RMSD (P atoms): {rmsd_val:.3f} Å")

    deltas = collect_torsion_differences(struct1, struct2)

    mcq_val = compute_mcq(deltas)
    print(f"MCQ (beta/gamma/delta angles): {mcq_val:.2f}°")

if __name__ == "__main__":
    main()
```

```
RMSD (P atoms): 14.985 Å
```

```
MCQ (beta/gamma/delta angles): 20.41°
```