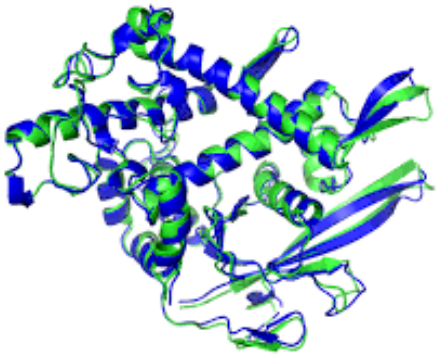
Introducción a la Bioinformática - UNQ
Prof. Dra. Ana Julia Velez Rueda

Trabajo práctico final: Analizador de Poros y Hidratación en Estructuras Proteicas

1 Objetivo

El objetivo de este trabajo práctico es desarrollar una biblioteca en Python que analice estructuras proteicas del PDB para identificar contactos proteína-ligando, detectar moléculas de agua de hidratación, y calcular la existencia de poros y posibles pasajes de agua a través de la estructura proteica.



2 Contexto

En el estudio de proteínas y su interacción con ligandos, el análisis de la interfaz de unión y el entorno hidratado es crucial para comprender los mecanismos moleculares de reconocimiento y unión. La hidratación juega un papel fundamental en la estabilización de complejos proteína-ligando y en la formación de canales de transporte. La detección automática de contactos y la caracterización de poros acuosos permite:

- Identificar sitios de unión críticos
- Analizar la especificidad de interacciones
- Estudiar mecanismos de transporte molecular
- Diseñar ligandos con mejor afinidad y selectividad

Actualmente, estos análisis requieren múltiples herramientas especializadas y procesamiento manual. Una biblioteca unificada que integre estas funcionalidades facilitará el análisis estructural sistemático de complejos proteína-ligando.

3 Requerimientos detallados

El software deberá mostrar:

1) Soporte para archivos CIF y PDB locales, y códigos PDB para descarga automática
2) Identificación automática de ligandos y moléculas de agua
3) Identificación automática de ligandos y moléculas de agua
4) Procesamiento de estructuras multiméricas

> **REQUISITOS OBLIGATORIOS:** El programa deberá tener un help bien detallado con los comandos y parámetros que pueden ser utilizados. El proyecto en GitHub deberá contar con una wiki de documentación y archivos y tutoriales de ejemplos de corrida. Construir una versión distribuída en paquete de CONDA[1] o PIP[2]. Tests unitarios.

Preprocesamiento

- Limpieza de estructuras (remoción de heteroátomos no esenciales)
- Identificación de cadenas proteicas y componentes
- Clasificación de moléculas de agua y ligandos

Algoritmo de detección de cavidades

- Implementación de métodos grid-based o esféricos
- Cálculo de volúmenes y dimensiones de poros
- Identificación de constricciones y aperturas

Análisis de conectividad

- Determinación de pasajes continuos de agua
- Cálculo de rutas posibles para moléculas de agua
- Evaluación de permeabilidad

Visualización

- Generación de representaciones estructurales con contactos destacados
- Visualización de redes de hidratación y poros
- Exportación de figuras en formatos estándar (PNG, PDF)

Tecnologías Sugeridas

- **BioPython**: Procesamiento de archivos PDB y análisis estructural
- **MDTraj**: Análisis de trayectorias y cálculos geométricos
- **NumPy/SciPy**: Cálculos numéricos y algoritmos
- **Matplotlib/Plotly**: Visualización 2D y 3D

[1] CONDA: https://docs.conda.io/en/latest/
[2] PIP: https://pypi.org/

- **PyMOL** (opcional): Integración para visualización avanzada

Algoritmos sugeridos:

- **HOLE**: Para caracterización de poros (implementación Python disponible)
- **DSSP**: Para asignación de estructura secundaria
- **Grid-based methods**: Para detección de cavidades

Especificaciones técnicas

- **Interfaz**: CLI con argparse y API Python
- **Formato de salida**: JSON estructurado con métricas calculadas

```
json
{
  "pdb_id": "1ABC",
  "contacts_analysis": {
    "ligand_id": "LIG",
    "contact_residues": ["ASP123", "GLU45", ...],
    "interaction_types": {"hydrogen_bonds": 5, "hydrophobic": 3, ...},
    "distances": {...}
  },
  "hydration_analysis": {
    "interface_waters": 15,
    "bridging_waters": 3,
    "water_network": {...}
  },
  "pore_analysis": {
    "pores_detected": true,
    "pore_volume": 150.5,
    "constriction_points": [...],
    "water_pathways": [...]
  },
  "visualization": {
    "contact_map": "contacts_1ABC.png",
    "pore_diagram": "pore_1ABC.png"
  }
}
```

- **Soportado**: Linux, macOS (Windows opcional)
- **Dependencias**: Gestión automática via requirements.txt o environment.yml

4 Forma de Entrega

El trabajo práctico se realizará en equipos de hasta 4 integrantes, y se entregará el día 8 de diciembre. El mismo deberá además estar en un repositorio público en Github (creado para la materia), el cual deberá contener un archivo README.md con los datos de l@s integrantes del equipo. Tod@s l@s integrantes del equipo deben tener commits con aportes significativos. Los trabajos serán presentados por los miembros del equipo en una exposición oral (con DEMO), la última clase de la cursada (10 de Diciembre). Los trabajos contarán con una calificación grupal por código y conceptual evaluando los contenidos de la materia. Esta nota contempla no sólo los aspectos de diseño del software y de apropiación de conceptos biológicos o de herramientas bioinformáticas, sino que también otros aspectos que se consideran importantes en la evaluación por proceso, como ser: entrega de versiones y consultas previas, indagación/investigación por fuera de lo propuesto por la

docente, entregas en los tiempos pautados, trabajo colaborativo/en equipo. Cada estudiante tendrá además una calificación individual correspondiente a los aportes y su proceso personal.

5 Material Educativo Suplementario

- CLI - Python: https://github.com/AJVelezRueda/Fundamentos_de_informatica/blob/master/CLI_con_Argparse/Interfaz_por_liena_de_comando.md
- Python Entornos: https://github.com/AJVelezRueda/Fundamentos_de_informatica/blob/master/Python_intro/ambientes%20Python/ambiente_python.md
- Estructura de Paquetes en Python: https://github.com/AJVelezRueda/ejemplo_setuptools_tox