University of Padova

Department of Mathematics "Tullio Levi-Civita"

Master Thesis in Data Science

Designing and Developing a Python Package for Analyzing Biological Data regarding Conformational Ensembles of Intrinsically Disordered Proteins.

Supervisor
Prof. Alexander Miguel Monzon
University of Padova

Master Candidate
Nikola Ivanović

Student ID
2105038

Academic Year
2023-2024

Abstract

Although many proteins require a well-defined structure for their function, a significant portion of an organism's proteome comprises intrinsically disordered regions (IDRs) that do not adopt a defined three-dimensional structure. Proteins with entirely disordered sequences, known as intrinsically disordered proteins (IDPs), exhibit functionality that diverges from the classical structure-function paradigm of globular proteins. Given the extensive conformational variability of IDPs, their diverse states are best represented by a structural ensemble, which consists of a set of conformations along with their corresponding statistical weights. To determine these structural ensembles, a range of experimental and computational methods are employed, often in combination to overcome the limitations of individual techniques. Among computational methods, molecular simulations using atomistic and coarse-grained models are particularly prominent. This has led to the creation of numerous software packages for molecular simulations, alongside a growing number of specialized stand-alone packages for simulation analysis. While these trajectory analysis tools may not be as robust as simulation engines, they provide a more flexible and customizable alternative. Despite the availability of various software for analyzing protein structures and structural ensembles of IDPs, there remains a notable shortage of tools offering efficient routines for comparing these ensembles. This thesis introduces IDPET (Intrinsically Disordered Protein Ensemble Toolkit), a Python software package developed to address this need. IDPET offers a wide range of analytical functions, including local analysis of residue-level features across all conformations within ensembles and global analysis for deriving conformation-level averages. It also provides ensemble comparison methods to assess differences between feature distributions and utilizes dimensionality reduction to visualize and cluster ensemble features, aiding in the identification of distinct conformational substates. The effectiveness of IDPET is demonstrated through a detailed analysis of three structural ensembles of the unfolded drkN SH_3 domain, derived from a fully random pool, an experimentally restrained pool, and a mixture of both.

Contents

Listing of figures

Listing of tables

Listing of acronyms

AF2 ..... Alphafold2

API ..... Application Programming Interface

CD ..... Circular Dichroism

cryo-EM ..... Cryogenic Electron Microscopy

DNA ..... Deoxyribonucleic Acid

DSSP ..... Dictionary of Secondary Structure in Proteins

EMD ..... Earth Mover's Distance

EPR ..... Electron Paramagnetic Resonance

FCS ..... Fluorescence Correlation Spectroscopy

FRET ..... Fluorescence Resonance Energy Transfer

FTIR ..... Fourier Transform Infrared

IDP ..... Intrinsically Disordered Protein

IDPET ..... Intrinsically Disordered Protein Ensemble Toolkit

IDR ..... Intrinsically Disordered Region

JSD ..... Jensen-Shannon Divergence

KLD ..... Kullback–Leibler Divergence

MC ..... Monte Carlo

MD ..... Molecular Dynamics

ML ..... Machine Learning

MM ..... Molecular Mechanics

MSA ..... Multiple Sequence Alignment

NMR . . . . . Nuclear Magnetic Resonance

NNP . . . . . Nearest-Neighbor Projection

PCA . . . . . Principal Component Analysis

PDB . . . . . Protein Data Bank

PED . . . . . Protein Ensemble Database

PFG-NMR . . . . . Pulsed Field Gradient NMR

QM . . . . . Quantum Mechanics

RMSD . . . . . Root-Mean-Square Deviation

SASA . . . . . Solvent Accessible Surface Area

SAXS . . . . . Small-Angle X-Ray Scattering

t-SNE . . . . . t-Distributed Stochastic Neighbor Embedding

UMAP . . . . . Uniform Manifold Approximation and Projection

1 Introduction

1.1 Protein Structure

Nearly every dynamic function of a living organism relies on proteins, which make up more than 50% of the dry mass of most cells and are crucial for almost all biological activities. Some proteins accelerate chemical reactions, while others are involved in defense, storage, transport, cellular communication, movement, or structural support.

The functional properties of proteins depend on their three-dimensional structures, which arise from the folding of specific amino acid sequences into compact domains. These folded domains act as building blocks for larger assemblies, such as virus particles or muscle fibers, and provide specific catalytic or binding sites, commonly seen in enzymes or proteins that bind oxygen or regulate DNA function. Therefore, understanding the biological function of proteins requires deducing or predicting their three-dimensional structure from their amino acid sequence. This task is exceedingly complex due to the 20 different amino acids that can combine into an almost infinite number of possible protein structures.

All amino acids share a common structure centered around a central carbon atom (Cα) , which is bonded to a hydrogen atom, an amino group (NH_2), and a carboxyl group (COOH) (Figure 1.1a). The unique feature of each amino acid is its side chain, also known as the R group, attached to the Cα atom. The R group can vary from a simple hydrogen atom to a complex carbon skeleton with various functional groups. The physical and chemical properties of the

side chain determine the unique characteristics of each amino acid, influencing its functional role in a polypeptide.

During protein synthesis, amino acids are linked end-to-end through peptide bonds, formed by condensing the carboxyl group of one amino acid with the amino group of the next, with the elimination of water (Figure 1.1b). This process repeats to elongate the polypeptide chain, where the amino group of the first amino acid and the carboxyl group of the last remain intact, defining the chain from its amino terminus to its carboxy terminus. This succession of peptide bonds creates a backbone from which various side chains project.

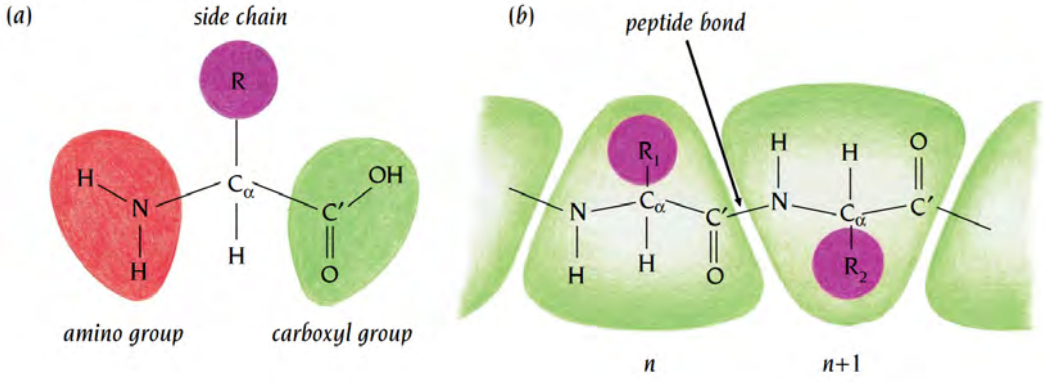

Figure 1.1: Proteins are constructed from amino acids linked by peptide bonds to form a polypeptide chain. Each amino acid (a) features a central carbon atom (Cα) bonded to an amino group (NH_2), a carboxyl group (COOH), a hydrogen atom (H), and a unique side chain (R). In the polypeptide chain (b), the carboxyl group of one amino acid forms a peptide bond with the amino group of the next, resulting in the elimination of a water molecule. This process repeats to create a chain of residues, where each residue shares a common main-chain structure (Cα, NH, C=O, H) and varies in its side chain (R) attached to the Cα atom. This figure was adapted from [1].

The 20 amino acids can be categorized into three distinct groups based on the chemical nature of their side chains. The first group includes amino acids with strictly hydrophobic side chains: Alanine (Ala), Valine (Val), Leucine (Leu), Isoleucine (Ile), Phenylalanine (Phe), Proline (Pro), and Methionine (Met). The second group is formed by charged residues: Aspartic acid (Asp), Glutamic acid (Glu), Lysine (Lys), and Arginine (Arg). The third group consists of amino acids with polar side chains: Serine (Ser), Threonine (Thr), Cysteine (Cys), Asparagine (Asn), Glutamine (Gln), Histidine (His), Tyrosine (Tyr), and Tryptophan (Trp). One exception to this classification is Glycine (Gly), which has only a hydrogen atom as its side chain. Due to its simplicity, Glycine possesses special properties and can be considered to belong either to a fourth group of amino acids or to the first group.

Peptide units are essentially rigid groups linked in a chain by covalent bonds at the Cα atoms.

Their only degrees of freedom are rotations around these specific bonds. Each unit can rotate around two bonds: the Cα-C' and the N-Cα bonds. These rotations are described by two angles: phi (ϕ) for the rotation around the N-Cα bond, and psi (ψ) for the rotation around the Cα-C' bond from the same Cα atom. Each amino acid residue is thus characterized by specific ϕ and ψ angles, which completely determine the conformation of the entire polypeptide backbone when defined accurately.

Despite their vast diversity, proteins share three fundamental levels of structure: primary, secondary, and tertiary. A fourth level, quaternary structure, is present when a protein is composed of two or more polypeptide chains (Figure 1.2). The primary structure of a protein refers to its specific sequence of amino acids, which is determined by genetic information. This primary structure dictates the protein's secondary and tertiary structures, influenced by the chemical properties of the amino acid backbone and side chains along the polypeptide. Most proteins have segments of their polypeptide chains that repeatedly coil or fold into patterns contributing to the protein's overall shape. These patterns, known as secondary structures, are formed by hydrogen bonds between the repeating units of the polypeptide backbone. One type of secondary structure is the alpha helix, a coil stabilized by hydrogen bonds between every fourth amino acid. Another type is the beta pleated sheet, where two or more polypeptide segments (beta strands) lie side by side, connected by hydrogen bonds between parallel segments. Tertiary structure, superimposed on the secondary structure patterns, represents the overall shape of a polypeptide. Unlike secondary structure, which involves backbone interactions, tertiary structure results from interactions between the amino acid side chains (R groups). Some proteins are made up of multiple polypeptide chains that combine into a single functional macromolecule. Quaternary structure describes the overall arrangement of these polypeptide subunits [14, 1].

1.2 Protein Structure Data

Protein structures are typically represented using Cartesian coordinates (x, y, z) of their constituent atoms in a standard format known as the Protein Data Bank (PDB) format. X-ray and cryo-EM structure files usually include coordinates for only the heavy atoms (C, N, O, and S), whereas NMR structure files generally contain coordinates for both the heavy atoms and the attached hydrogen atoms.

The PDB format (Figure 1.3) is a machine- and human-readable format that contains detailed information about a protein structure, including data about the protein itself, the depositors, the sequence, the secondary structure, and the Cartesian coordinates of the atoms. Each

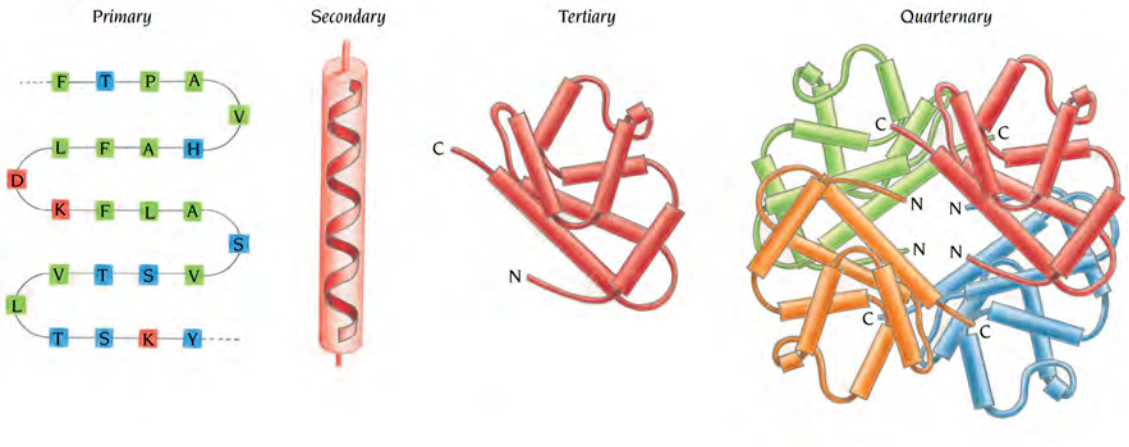

Figure 1.2: The primary structure of a protein is its amino acid sequence. Specific regions in this sequence form secondary structures like alpha helices and beta strands. These secondary structures fold into a tertiary structure, comprising compact globular units called domains. Some proteins have multiple polypeptide chains, forming a quaternary structure. This figure was taken from [1].

line in a PDB file begins with a specific seven-letter tag, followed by a fixed number of spaces and the corresponding information. Typically, each PDB file follows a consistent structure.

At the beginning of the document, the HEADER denotes the protein function, PDB ID, and deposition date; CMPND indicates the protein name; and SOURCE specifies the source organism. The AUTHOR section lists the authors of the structure, while the JRNL section lists the journals that published the structure.

Following these sections, the REMARK tag contains additional details such as the resolution, R factor (a measure of quality), methods used to solve the structure, and the number of molecules in the asymmetric unit. The amino acid sequence is signified by the SEQRES tag, given by three-letter amino acid codes. The HET and FORMUL labels signify the names and chemical formulae of hetero-atoms. Finally, the HELIX, SHEET, TURN, and SSBOND tags denote secondary structure elements.

Following these tags that together constitute the header of a PDB file, the subsequent lines provide the actual atomic coordinates and are denoted by the ATOM tag. Each line includes the atom number, atom label (such as CA for alpha carbon or C for carbonyl carbon), residue name (three-letter code), chain identifier, residue number, X coordinate (in ångstroms), Y coordinate (in ångstroms), Z coordinate (in ångstroms), occupancy (typically 1.00), and the thermal B factor (a measure of atomic mobility) [2].

```
HEADER    ELECTRON TRANSPORT                      19-MAR-90   2TRX      2TRXA  1
COMPND    THIOREDOXIN                                                   2TRXA  2
SOURCE    (ESCHERICHIA $COLI)                                           2TRX   4
AUTHOR    S.K.KATTI,D.M.LE*MASTER,H.EKLUND                              2TRX   5
JRNL        AUTH   S.K.KATTI,D.M.LE*MASTER,H.EKLUND                     2TRX   7
JRNL        TITL   CRYSTAL STRUCTURE OF THIOREDOXIN FROM ESCHERICHIA    2TRX   8
JRNL        TITL 2 $COLI AT 1.68 ANGSTROMS RESOLUTION                   2TRX   9
JRNL        REF    J.MOL.BIOL.                   V. 212   167 1990      2TRX  10
JRNL        REFN   ASTM JMOBAK  UK ISSN 0022-2836                 070   2TRX  11
REMARK   2                                                              2TRX  31
REMARK   2 RESOLUTION. 1.68 ANGSTROMS.                                  2TRX  32
REMARK   3                                                              2TRX  33
REMARK   3 REFINEMENT. BY THE RESTRAINED LEAST-SQUARES PROCEDURE OF J.  2TRX  34
REMARK   3  KONNERT AND W. HENDRICKSON AS MODIFIED BY B. FINZEL         2TRX  35
REMARK   3  (PROGRAM *PROFFT*).  THE R VALUE IS 0.165 FOR 25969         2TRX  36
REMARK   3  REFLECTIONS IN THE RESOLUTION RANGE 8.0 TO 1.68 ANGSTROMS   2TRX  37
REMARK   3  WITH FOBS .GT. 3.0*SIGMA(FOBS)                              2TRX  38
REMARK   3                                                              2TRX  39
SEQRES   1 A  108  SER ASP LYS ILE ILE HIS LEU THR ASP ASP SER PHE ASP  2TRX  74
SEQRES   2 A  108  THR ASP VAL LEU LYS ALA ASP GLY ALA ILE LEU VAL ASP  2TRX  75
SEQRES   3 A  108  PHE TRP ALA GLU TRP CYS GLY PRO CYS LYS MET ILE ALA  2TRX  76
SEQRES   4 A  108  PRO ILE LEU ASP GLU ILE ALA ASP GLU TYR GLN GLY LYS  2TRX  77
SEQRES   5 A  108  LEU THR VAL ALA LYS LEU ASN ILE ASP GLN ASN PRO GLY  2TRX  78
SEQRES   6 A  108  THR ALA PRO LYS TYR GLY ILE ARG GLY ILE PRO THR LEU  2TRX  79
SEQRES   7 A  108  LEU LEU PHE LYS ASN GLY GLU VAL ALA ALA THR LYS VAL  2TRX  80
SEQRES   8 A  108  GLY ALA LEU SER LYS GLY GLN LEU LYS GLU PHE LEU ASP  2TRX  81
SEQRES   9 A  108  ALA ASN LEU ALA                                       2TRX  82
HET    MPD    606       8     2-METHYL-2,4-PENTANEDIOL                  2TRX 107
HET    MPD    607       8     2-METHYL-2,4-PENTANEDIOL                  2TRX 108
HET    MPD    608       8     2-METHYL-2,4-PENTANEDIOL                  2TRX 109
FORMUL   3   CU    2(CU1 ++)                                            2TRX 110
FORMUL   4  MPD    8(C6 H14 O2)                                         2TRX 111
FORMUL   5  HOH   *140(H2 O1)                                           2TRX 112
HELIX    1 A1A SER A   11  LEU A   17  1 DISORDERED IN MOLECULE B       2TRX 113
HELIX    2 A2A CYS A   32  TYR A   49  1 BENT BY 30 DEGREES AT RES 39   2TRX 114
HELIX    3 A3A ASN A   59  ASN A   63  1                               2TRX 115
HELIX    4 31A THR A   66  TYR A   70  5 DISTORTED H-BONDING C-TERMINS  2TRX 116
HELIX    5 A4A SER A   95  LEU A  107  1                               2TRX 117
HELIX    6 A1B SER B   11  LEU B   17  1 DISORDERED IN MOLECULE B       2TRX 118
SSBOND   1 CYS A   32    CYS A   35                                     2TRX 143
ATOM      1  N   SER A   1      21.389  25.406  -4.628  1.00 23.22      2TRX 152
ATOM      2  CA  SER A   1      21.628  26.691  -3.983  1.00 24.42      2TRX 153
ATOM      3  C   SER A   1      20.937  26.944  -2.679  1.00 24.21      2TRX 154
ATOM      4  O   SER A   1      21.072  28.079  -2.093  1.00 24.97      2TRX 155
ATOM      5  CB  SER A   1      21.117  27.770  -5.002  1.00 28.27      2TRX 156
ATOM      6  OG  SER A   1      22.276  27.925  -5.861  1.00 32.61      2TRX 157
ATOM      7  N   ASP A   2      20.173  26.028  -2.163  1.00 21.39      2TRX 158
ATOM      8  CA  ASP A   2      19.395  26.125  -0.949  1.00 21.57      2TRX 159
ATOM      9  C   ASP A   2      20.264  26.214   0.297  1.00 20.89      2TRX 160
ATOM     10  O   ASP A   2      19.760  26.575   1.371  1.00 21.49      2TRX 161
ATOM     11  CB  ASP A   2      18.439  24.914  -0.856  1.00 22.14      2TRX 162
ATOM     22  CE  LYS A   3      21.620  21.104   2.844  1.00 25.84      2TRX 173
ATOM     23  NZ  LYS A   3      20.830  20.757   1.615  1.00 25.55      2TRX 174
```

Figure 1.3: An example of a Protein Data Bank (PDB) formatted file displaying the initial lines of the Escherichia coli thioredoxin entry (PDB ID: 2TRX). This figure was taken from [2].

1.3 Intrinsically Disordered Proteins

Traditionally, protein function has been viewed as dependent on the well-defined, folded three-dimensional structure of the polypeptide chain. The classical structure-function paradigm (Figure 1.4; left panel) suggests that a protein's sequence defines its structure, which in turn determines its function.

Classification schemes help systematically assign functions to proteins, typically consisting of one or multiple structured domains. These domains fold independently, form precise tertiary contacts, and adopt specific three-dimensional structures to perform their functions. The sequences composing structured domains can be grouped into families of homologous sequences that share common evolutionary relationships and molecular functions.

While many proteins require a well-defined structure to function, a large fraction of an organism's proteome consists of polypeptide segments that do not form a defined three-dimensional structure. These segments are known as intrinsically disordered regions (IDRs; Figure 1.4; right panel). Consequently, their functionality differs from the classical structure-function view of globular proteins. According to the disorder-function paradigm, protein sequences in a genome are modular, combining both structured and disordered regions (Figure 1.4; bottom panel). Proteins lacking IDRs are classified as structured proteins, whereas proteins with entirely disordered sequences that do not adopt any defined tertiary structure are termed intrinsically disordered proteins (IDPs) [3, 15].

The ability of polypeptide segments to fold into a defined tertiary structure cohesively is largely influenced by long-range hydrophobic interactions among amino acids along the linear sequence. In contrast, IDRs lack sufficient hydrophobic amino acids to facilitate cooperative folding. Instead, IDRs typically consist of a higher proportion of polar or charged amino acids. Consequently, IDRs do not adopt a distinct three-dimensional structure either entirely or in parts in their native state. They tend to sample a range of conformations that are in dynamic equilibrium under physiological conditions [16].

Due to remarkable progress in comparative evolutionary and experimental structure-function research, it is now evident that structural disorder offers numerous functional benefits. IDP functions either originate directly from their disorder (as entropic chains) or from molecular recognition, where they undergo induced folding (disorder-to-order transition) upon binding to a partner molecule. Several specific functional modalities, including adaptability in binding, high functional density, weak yet specific binding, and frequent regulation by post-translational modification, have been clearly demonstrated.

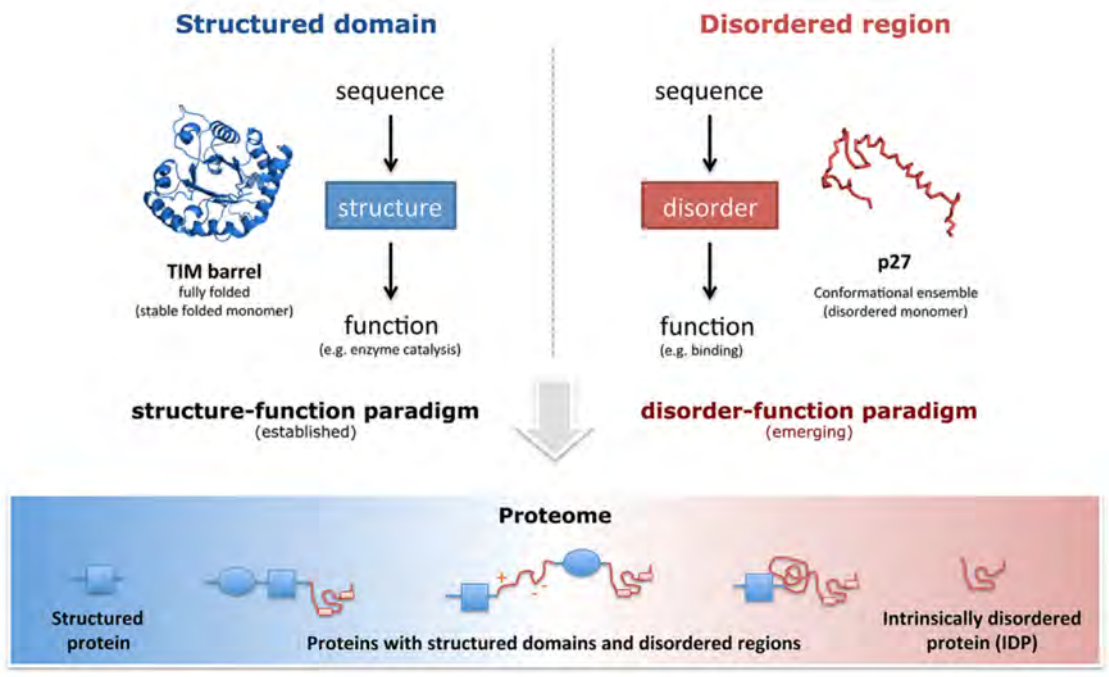

Figure 1.4: Structured domains and intrinsically disordered regions (IDRs) are two fundamental classes of functional building blocks of proteins. The synergy between disordered regions and structured domains increases the functional versatility of proteins. This figure was taken from [3].

From a biological standpoint, the role of structural disorder is crucial in determining which cellular functions benefit most from the absence of a stable protein structure. This question has been extensively investigated in large-scale bioinformatics studies, highlighting that IDPs predominantly play roles in signaling and regulation processes. Functions associated with long disordered regions frequently correlate with regulatory mechanisms involved in transcription and translation. In contrast, functions correlated with structured proteins often center around enzymatic catalysis.

Structural disorder is prevalent across all species, with a notably higher presence in eukaryotes compared to prokaryotes due to its strong association with regulatory and signaling functions. Conservative estimates suggest that 10–35% of prokaryotic proteins and 15–45% of eukaryotic proteins contain significant disorder, defined as long disordered regions of at least 30 residues. While it is commonly accepted that structural disorder increases with biological complexity, the highest levels are not found in the most complex metazoan eukaryotes (such as humans), but rather in single-celled eukaryotes that exhibit a host-changing lifestyle [17, 18, 19].

Considering the significant conformational variability exhibited by IDPs, their diverse states

are best represented through structural ensembles. By definition, a structural ensemble is a set of conformations together with corresponding statistical weights. The statistical weights define the likelihood of encountering a particular conformation inhabited by a protein under well-defined experimental conditions [8].

1.4 Structural Ensemble Determination

There has been increasing interest in both experimental and computational approaches to accurately capture structural ensembles of IDPs.

Experimental methods such as nuclear magnetic resonance (NMR), small-angle X-ray scattering (SAXS), and single-molecule spectroscopy have proven highly valuable for gathering structural information. Techniques like fluorescence resonance energy transfer (FRET), fluorescence correlation spectroscopy (FCS), and SAXS provide measurements of overall chain dimensions, while NMR spectroscopy offers site-specific details, including secondary structure content, distances between labeled sites, and hydrodynamic radius measurements through pulsed field gradient NMR (PFG-NMR).

Computational methods typically leverage theoretical models of physical and chemical interactions within the system to generate structural ensembles. These methods can be broadly categorized into two types: (1) those that incorporate experimental data to guide ensemble generation or selection, and (2) those that generate IDP ensembles de novo, without experimental input.

In the first category, one common approach is to use experimental data as restraints in simulations. Another approach involves statistical methods, such as ENSEMBLE [20], EOM [21], and ASTEROIDS [22], which utilize experimental data to select ensembles from pre-generated conformational pools. Additionally, ensembles consistent with experimental data can also be selected from conformations generated via molecular dynamics (MD) simulations.

For the second category, a range of simulation techniques—including MD, Monte Carlo (MC), metadynamics, and replica exchange—along with different levels of representation (coarse-grained, implicit solvent, and all-atom with explicit water) have been applied to generate IDP ensembles de novo [23].

Recent advances in machine learning techniques like AlphaFold2 [24] and RoseTTAFold [25] have significantly transformed protein structure prediction. While these models were primarily developed to predict structures of folded proteins, several pipelines have been proposed to extend their application to predicting multiple conformational states. Moreover, new ma-

chine learning approaches specifically designed for generating structural ensembles, such as idpGAN [11], have been inspired by these breakthrough models.

1.4.1 Experimental Methods

Disordered protein states are highly heterogenous and dynamic, making them unaccessible to X-ray crystalography. Although various solution spectroscopy methods, such as circular dichroism (CD), fluorescence, Fourier transform infrared (FTIR), and electron paramagnetic resonance (EPR) have been utilized, two techniques stand out for their contribution to structural data: NMR spectroscopy, which offers atromic-level information on secondary structure, tertiary interactions, solvent exposure, overall molecular dimensions and dynamic properties, and SAXS, a highly specialized method used to measure molecular dimensions [26].

NMR Spectroscopy

In NMR spectroscopy, protein structures are determined by measuring the absorption of radio waves by atomic nuclei, such as hydrogen (^{1}H), isotopically labeled carbon (^{13}C), or nitrogen (^{15}N). This absorption measurement indicates the transfer of nuclear magnetism from one atom to another. In NMR, this magnetization transfer is measured through chemical shifts, J-couplings, and nuclear Overhauser effects (NOEs). These measurements provide a set of approximate structural constraints that can be input into a computer-based constraint minimization calculation. The output is a series of similar protein structures that satisfy the experimental constraints. These structures are then overlaid or superimposed on each other to create blurrograms. The quality of an NMR structure determination is typically assessed by how closely these superimposed structures match, with root mean square deviation (RMSD) values of less than 1 Å indicating a high-quality structure and RMSD values greater than 2 Å indicating a poorly determined structure [2].

Chemical Shifts (CSs) are the most straightforward parameter to obtain from an NMR experiment. CSs reflect how nuclear magnetic energy levels vary depending on the electronic environment around a nucleus. Due to their inherent flexibility, IDPs typically exhibit CSs close to random-coil values. Deviations from these values, especially for backbone carbon atoms, suggest the presence of secondary structural elements. For a ^{15}N-labeled protein, the $^{1}H\ -^{15}N\ HSQC$ (heteronuclear single quantum coherence) experiment produces peaks representing the N–H correlations for each amino acid in the protein. These peaks serve as the NMR "fingerprint" of the protein [27]. NMR spectra of IDPs are characterized by a low (^{1}H)

signal dispersion reflecting the similar electronic environment experienced by the different protons (Figure 1.5) [4].

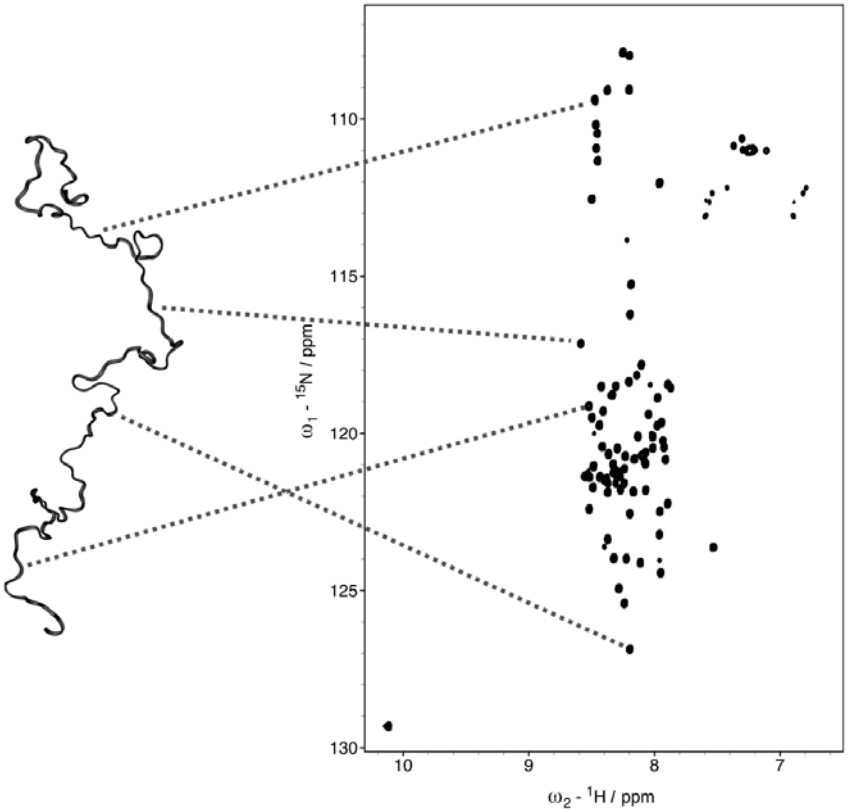

Figure 1.5: $^{1}H-^{15}N$ heteronuclear single quantum coherence (HSQC) spectrum of the intrinsically disordered C-terminal domain of the Sendai virus nucleoprotein, demonstrating the low signal dispersion in the ^{1}H dimension typical of an IDP. NMR spectroscopy offers site-specific information, with each resonance in the HSQC spectrum corresponding to a particular amide group in the protein. This figure was taken from [4].

Small Angle X-ray Scattering (SAXS)

Small angle X-ray scattering (SAXS) is a powerful technique used to study the structure of proteins in solution, regardless of whether they are ordered or disordered. It provides information on the size and shape of proteins and their complexes across a wide range of molecular sizes and under various experimental conditions.

A SAXS experiment (Figure 1.6a) is conceptually straightforward: a solution containing particles, typically held in a quartz capillary, is exposed to a focused, monochromatic X-ray beam. An X-ray detector records the intensity of the scattered X-rays. The scattering pattern of the

pure solvent is measured and subtracted from that of the sample solution to isolate the scattering signal from the particles of interest. The resulting scattering pattern provides information about the overall shape and size of the particles being studied.

Because the particles in solution are randomly oriented, the scattering pattern is isotropic. As a result, the pattern recorded by a two-dimensional detector can be averaged radially. The scattering intensity $I(s)$ is expressed as a function of the momentum transfer $s = \frac{4\pi \sin\theta}{\lambda}$, where λ is the wavelength of the beam and 2θ is the scattering angle. These isotropic patterns are represented as radially averaged one-dimensional curves $I(s)$ (Figure 1.6b). For flexible macromolecules, like IDPs, the measured intensity reflects a population-weighted average of the scattering curves from all the particles in the solution [5].

Key structural details, including the size, shape, compactness, and oligomeric state of a protein, can be obtained directly from a SAXS curve. The Kratky plot (Figure 1.6c), which graphs $[\mathrm{I(s)} \cdot s^2]$ against s, is commonly used to differentiate between globular proteins—characterized by a bell-shaped curve—and disordered proteins—showing a continuous increase with s. The radius of gyration R_g, obtained from the initial segment of the curve using Guinier's method, estimates the overall size of the particle [27].

1.4.2 Statistical Methods

Statistical methods for determining structural ensembles rely on generating a random pool of conformations, and subsequently applying algorithms to select subsets that correspond to experimental data. It is assumed that the experimental data related to the studied protein can be represented as an average over multiple conformers, the exact number of which is generally unknown. This experimental data can be obtained from methods such as NMR, SAXS, or a combination of various techniques.

Initially, a large and diverse pool of conformations is randomly generated by simulation software like TraDES [28] to cover the conformational space of the protein. A Monte Carlo-based search algorithm is then typically used to sample subsets of conformations that align with the experimental data. The algorithm iteratively samples subsets from the pool and compares their properties to the experimental observations, aiming to find an ensemble whose averaged properties match the experimental data.

Once an ensemble fitting all the experimental data is selected, quantitative criteria such as *Cα-Cα* distances, radius of gyration, and secondary structure distribution are used to analyze and evaluate the ensemble [20, 21, 22].

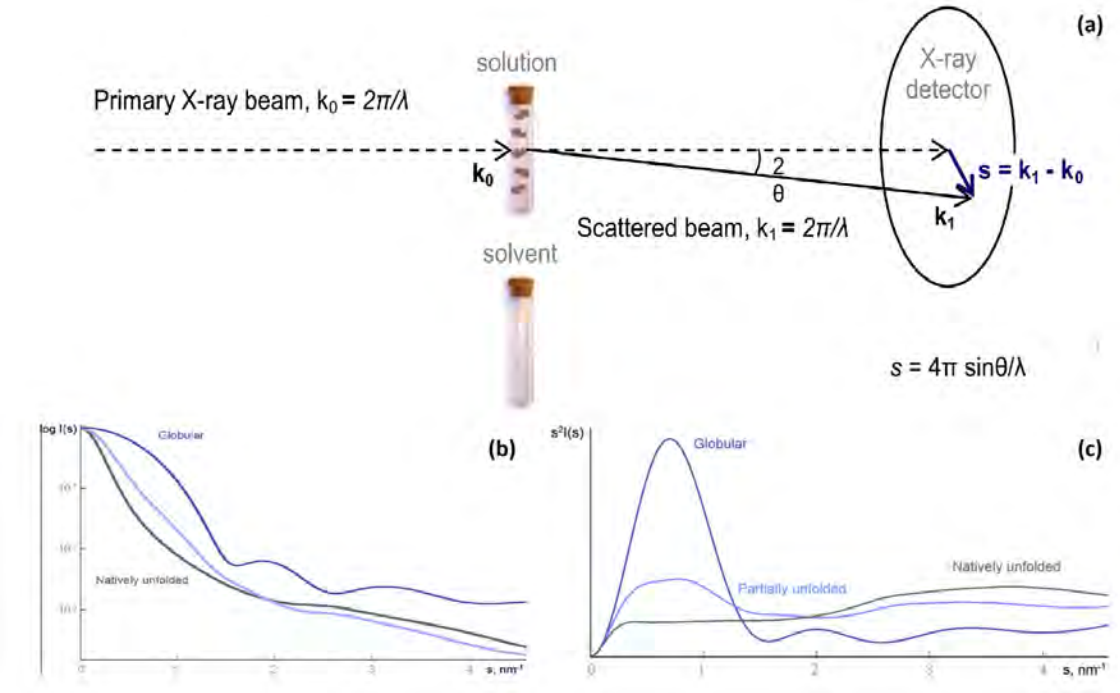

Figure 1.6: Small Angle X-ray Scattering (SAXS). a) Schematic representation of a SAXS experiment. (b) Logarithmic plot of the scattering intensity $I(s)$ (in arbitrary units) vs. s (in inverse nanometers). (c) Kratky plot of $s^2I(s)$ vs. s. Data simulated from three 60 kDa proteins: globular (dark blue), partially unfolded (light blue), and fully disordered (gray). This figure was adapted from [5].

1.4.3 Molecular Simulations

Molecular simulation techniques play a crucial role in understanding and predicting the properties, structure and function of molecular systems. The essence of any molecular simulation method is to create a particle-based model of the system and then use deterministic or probabilistic rules to generate a trajectory that describes its evolution during the simulation.

Molecular simulation methods can be divided into Molecular Dynamics (MD) and Monte Carlo (MC). MD integrates equations of motion to create a dynamical trajectory, useful for studying structural, dynamic, and thermodynamic properties. MC uses probabilistic rules to generate a sequence of states for calculating structural and thermodynamic properties, but not dynamic properties, as it lacks a concept of time.

Molecular simulation methods can use different physical theories to describe the particle-based model of the system (Figure 1.7). Quantum mechanics (QM) represents electrons explicitly in the model and calculates interaction energy by solving the electronic structure with minimal empirical parameters, though it involves various approximations. Molecular mechan-

ics (MM) represents molecules with particles for atoms or groups of atoms, assigning electric charges and empirical potential energy functions to calculate interactions.

MM simulations are much faster than quantum simulations, making them the preferred method for most biomolecular studies in the condensed phase. However, they are generally less accurate than QM simulations and cannot simulate bond rearrangements. Generally, QM simulations are feasible for systems with hundreds of atoms or fewer, while MD simulations can handle tens or hundreds of thousands of atoms [29].

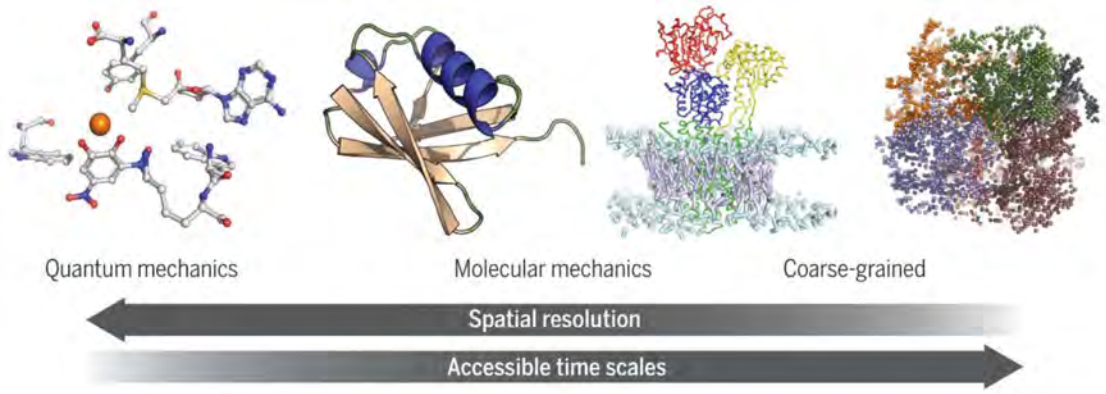

Figure 1.7: Computational approaches to studying biomolecules range from detailed quantum mechanical models to atomistic molecular mechanics to coarse-grained models, where several atoms are grouped together. The decreased computational complexity granted by progressive coarse-graining makes it possible to access longer time scales and greater length scales. This figure was taken from [6].

Despite advances in computing power, all-atom MD simulations remain limited to small systems and fast processes. Coarse-grained models offer greater computational efficiency, allowing simulations of larger systems and longer time scales. Well-designed coarse-grained models can be reconstructed to all-atom resolution, enabling multiscale modeling that combines the speed of coarse-grained simulations with the accuracy of all-atom MD.

Coarse-grained protein models use various levels of simplified polypeptide chain representation. The protein main chain can be represented by all heavy atoms or by one or two united atoms per residue, while the side chain is typically replaced by one or two united atoms.

In comparison to all-atom models, coarse-grained models smooth out the energy landscape (Figure 1.8). This smoothing effect helps prevent the model from getting trapped in local energy minima, allowing for more efficient exploration in search of the most likely structures, represented by global minima. Various definitions of interaction models for coarse-grained representations are possible. Physics-based derivations of coarse-grained force fields start from

classical all-atom interaction models and translate them into united atom potentials. In contrast, knowledge-based interaction schemes are derived from statistical regularities observed in known protein structures [7, 6].

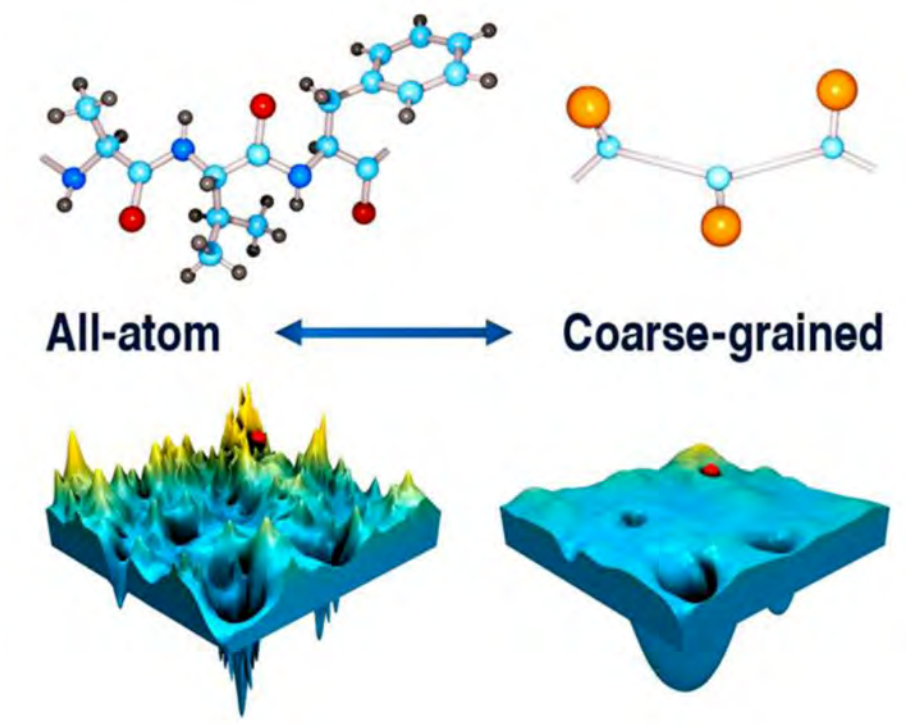

Figure 1.8: All-atom versus coarse-grained energy landscape. The figure illustrates the effect of the smoothening of the energy landscape in a coarse-grained model as compared to an all-atom model. The flattening enables efficient exploration of the energy landscape in search for the global minima, while avoiding traps in the local minima. This figure was taken from [7].

1.4.4 Combining Experimental and Theoretical Methods

Determining protein structural ensembles through experimental methods alone faces several challenges. Ideally, experimental techniques would distinguish heterogeneous states by having faster observation times than the interconversion between states. However, this is often not the case, resulting in averaged observations over multiple states. Moreover, experimental methods typically measure specific properties, providing sparse structural information about protein conformational fluctuations. Additionally, experimental observations are affected by random errors, stemming from statistical fluctuations, and systematic errors, which can arise from instrument faults, misuse, incorrect data assignment, or poor sample preparation.

Computational methods range from highly-detailed ab initio methods and all-atom empirical force fields to coarse-grained approaches. Despite their detail and accuracy, these models are still approximations of actual interatomic interactions and cannot fully predict all system properties. Additionally, limited computational resources restrict simulations to finite timescales (e.g., microseconds), which are often shorter than the timescales of the biological processes of interest.

Since neither experimental nor computational approaches can by themselves generate accurate structural ensembles, combined approaches have been developed to overcome the limitations of individual techniques (Figure 1.9). These approaches use experimental data as constraints for computational models, typically reflecting averages across all conformations within an ensemble. However, the number of degrees of freedom in the ensembles significantly exceeds the available experimental restraints, resulting in multiple potential solutions without a clear "best" option [8].

In fact, an infinite number of different conformational ensembles can be reconstructed from experimental data. Any reconstruction can be seen as a probability distribution over the space of allowable states, whether defined by Cartesian coordinates of atom nuclei, backbone dihedral angles, or Euler transformations for rigid protein domains.

To address this challenge, two primary approaches have been developed. One approach seeks to explain the data while maintaining maximum uncertainty, ensuring no additional assumptions about the unknown distribution are made. This solution satisfies the Maximum Entropy Principle. Solutions based on the Maximum Entropy Principle typically consist of a large number of individual states with small, often equal, probabilities.

The second approach seeks probability distributions where high probabilities are concentrated in a small number of states, identifying which conformations can carry significant weights. Various methods fall under this category and are collectively referred to as large weight solutions. The effectiveness of these solutions depends on the maximum number of states considered in the ensemble, with each method adopting a specific strategy to balance the number of conformers and the accuracy of the solution [30].

1.4.5 Machine Learning Approaches

Predicting protein structures from amino acid sequences has been a key area of scientific research for many decades due to its critical importance and well-defined physical and computational principles. Although progress has varied over time, there have been remarkable ad-

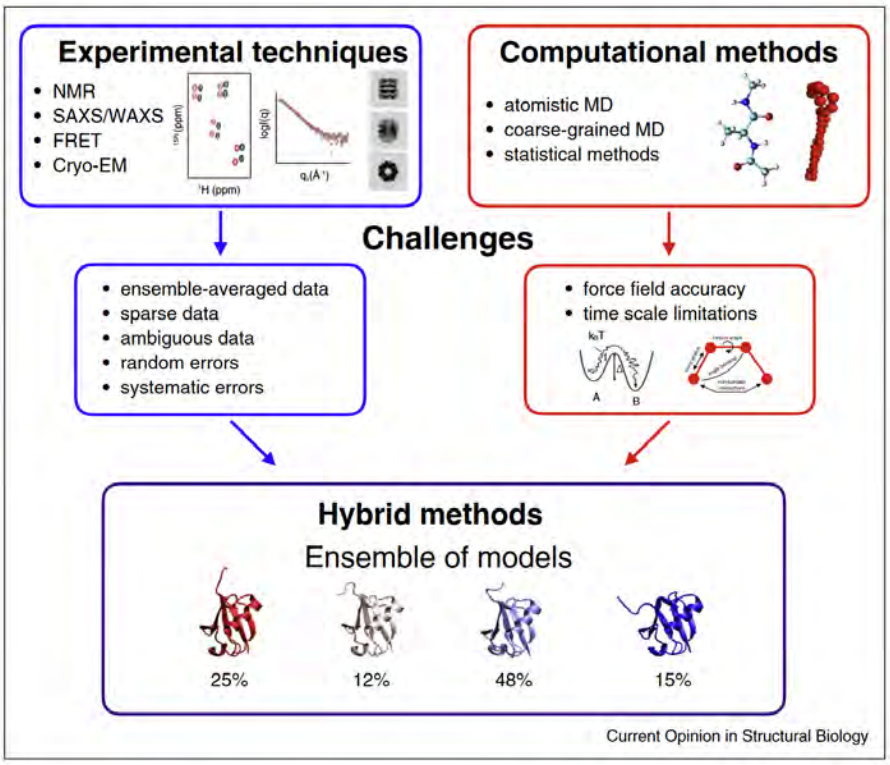

Figure 1.9: Determination of protein structural ensembles by combining experimental and theoretical methods. Experimental methods, such as NMR spectroscopy, SAXS/WAXS, and cryo-EM, typically provide ensemble-averaged, sparse, and sometimes ambiguous data, influenced by random and systematic errors. Computational methods, like molecular dynamics (MD) simulations, are limited by inaccuracies in force fields and accessible timescales. Combining experimental and computational techniques can overcome these limitations, leading to accurate determination of protein structural ensembles. This figure was taken from [8].

vancements in recent years. These breakthroughs are largely due to the increasing use of neural networks in structure prediction pipelines, replacing traditional energy models and sampling procedures [31].

Remarkable progress has been achieved thanks to coevolutionary methods, with DeepMind's AlphaFold2 being the most influential, as it arguably solves the single apo domain protein structure prediction problem. While these methods excel at determining the structures of folded proteins, modifications are often needed to address the structural variability exhibited by IDPs.

Alphafold2

Alphafold2 (AF2), a cutting-edge model from DeepMind, has been notably successful in predicting protein structures from amino acid sequences. Its performance was highly evaluated

during the 14th Critical Assessment of Protein Structure (CASP) experiment, accurately predicting 98.5% of the proteins in the human proteome. Predictions made using AF2 are now accessible via a portal hosted by the European Bioinformatics Institute. This breakthrough provides detailed, atomic-level models for most human proteins, showcasing the significant advancements achieved through curated knowledge bases and extensive sequencing data.

AF2 integrates physical and biological knowledge about protein structure in its deep learning algorithm design. The network directly predicts the 3D coordinates of all heavy atoms in a given protein using three main inputs: the primary amino acid sequence, multiple sequence alignments (MSAs) of evolutionarily related proteins, and the 3D atom coordinates of a small number of homologous structures (templates), when available. One key factor impacting the accuracy of AF2 predictions is recycling, which involves iterative refinement through the networks. In addition to the predicted coordinates, AF2 computes a per-residue confidence accuracy of the structure called the predicted local-distance difference test (pLDDT).

Interestingly, AF2 predictions emphasize the significance of IDPs/IDRs. A notable aspect of AF2's structural annotations of the human proteome is the large number of low and very low confidence regions that coincide with predicted IDRs. This underscores the importance of conformational heterogeneity. The annotations from AF2 provide a crucial boost for functional and biophysical research on IDPs/IDRs. A typical annotation, as illustrated in Figure 1.10, displays extensive regions marked in orange as "unstructured" with very low confidence [9].

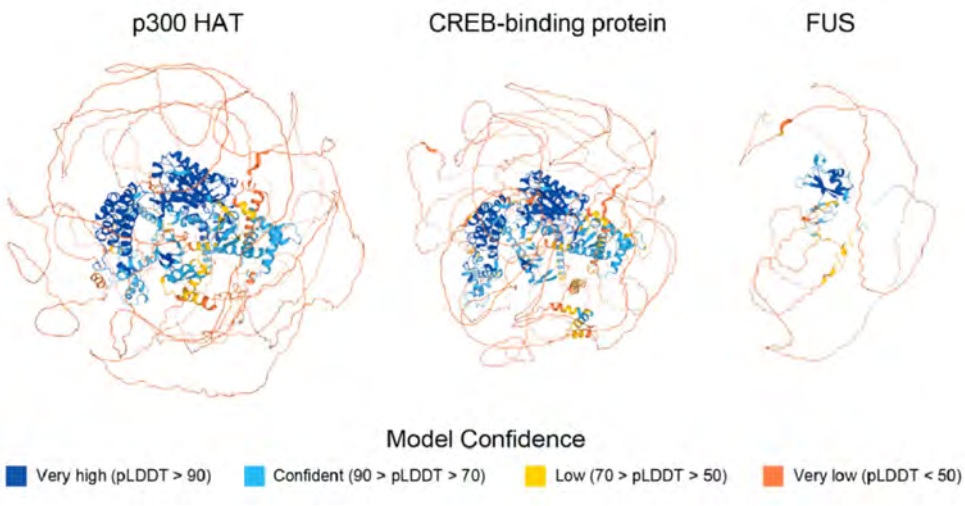

Figure 1.10: Example showing Alphafold2 predicted structures of proteins with long IDRs. Unstructured regions are depicted in orange to signify very low confidence. This figure was taken from [9].

Usually, AF2 models exhibit restricted conformational diversity. To overcome this, various pipelines have been suggested to augment the structural variability of the predicted ensemble using different strategies (Figure 1.11). One method [32] involves reducing the number of sequences employed in creating the MSA, alongside retaining templates and recycling. By selecting a smaller, less comprehensive MSA from the pool of homologous sequences at random, uncertainty is heightened, thereby improving sampling. An alternative method, known as SPEACH_AF [33], masks columns instead of rows to expand AF2 conformational sampling. This technique fundamentally changes the information within the MSA by replacing residues with alanine along the columns, thereby obscuring coevolutionary information. In an alternative pipeline [34], MSA sequences are clustered based on similarity, and each cluster is subsequently used as an MSA input to potentially generate models with different conformational states [10].

IdpGAN

Although machine learning methods like AF2 have achieved high accuracy in predicting conformations from amino acid sequences, these methods overlook the dynamic nature of proteins and focus on deriving a single conformational state. In contrast, IdpGAN [11] directly models conformational ensembles using a generative approach. It achieves this by learning probability distributions of conformations from a training dataset obtained from MD simulations. The output of this approach consists of Cartesian coordinates of conformations at the *Cα* coarse-grained level.

IdpGAN leverages a Generative Adversarial Network (GAN) to capture the distributions of conformations in the training set. The GAN comprises two neural networks: a generator (G) and a discriminator (D), which engage in an adversarial game (Figure 1.12). The G network in IdpGAN employs a transformer architecture that takes a randomly sampled latent sequence as input. This sequence is refined progressively through transformer blocks and is ultimately outputted as Cartesian coordinates for the *Cα* atoms. Additionally, the network requires a one-hot encoded amino acid sequence that provides essential conditional information for different proteins. IdpGAN also includes a self-attention mechanism that updates each residue's embedding with information from the entire sequence.

The D network's role during GAN training is to guide G toward generating data consistent with the training set. In IdpGAN, the D network inputs a protein conformation and an amino acid sequence, and outputs a scalar value representing the probability that the combination is valid. The conformation input is in the form of an intratomic distance matrix computed from

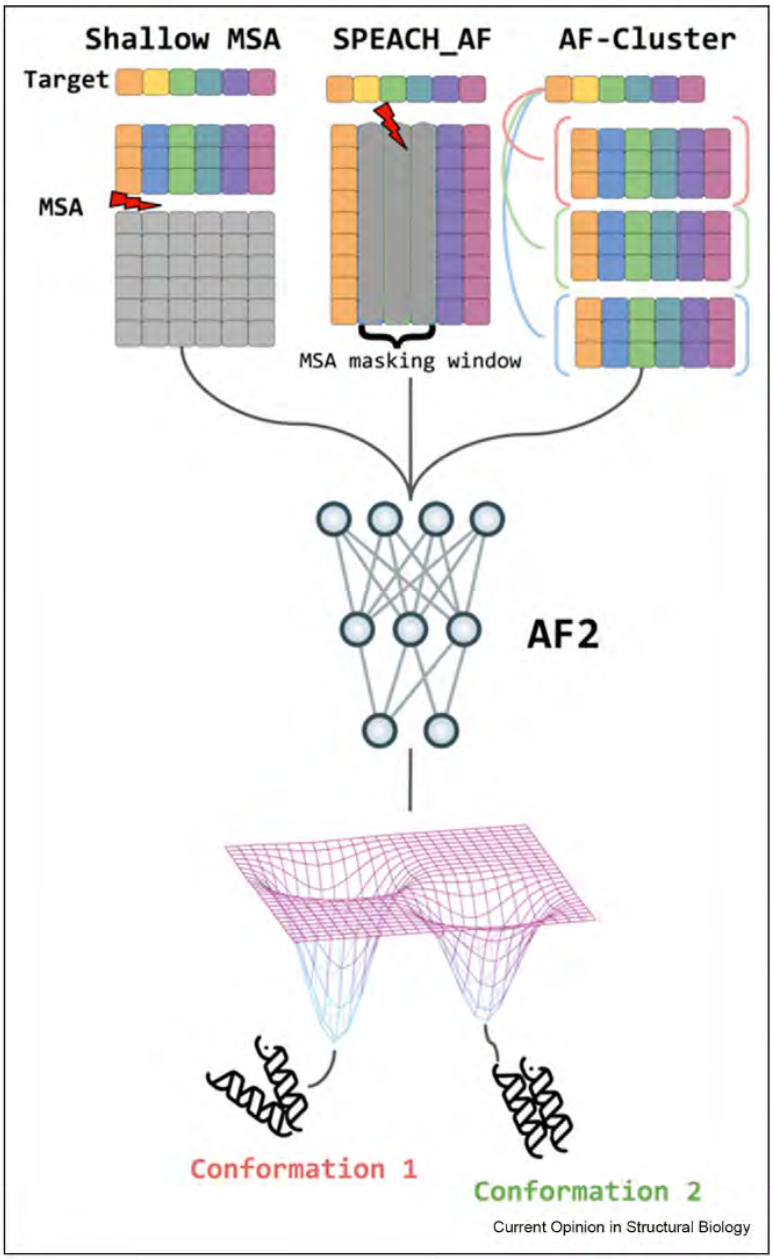

Figure 1.11: AF2 pipelines enhance the exploration of predicted conformational variability. A shallow multiple sequence alignment (MSA) extracts coevolutionary insights linked to various conformational states from a subset of randomly selected sequences. SPEACH_AF reveals alternative states by selectively masking the primary coevolutionary signal, thereby sampling different conformations. AFcluster utilizes sequence clustering to extract information pertaining to distinct conformational states. This figure was taken from [10].

coordinates, ensuring that IdpGAN is invariant to translations, rotations, and reflections.

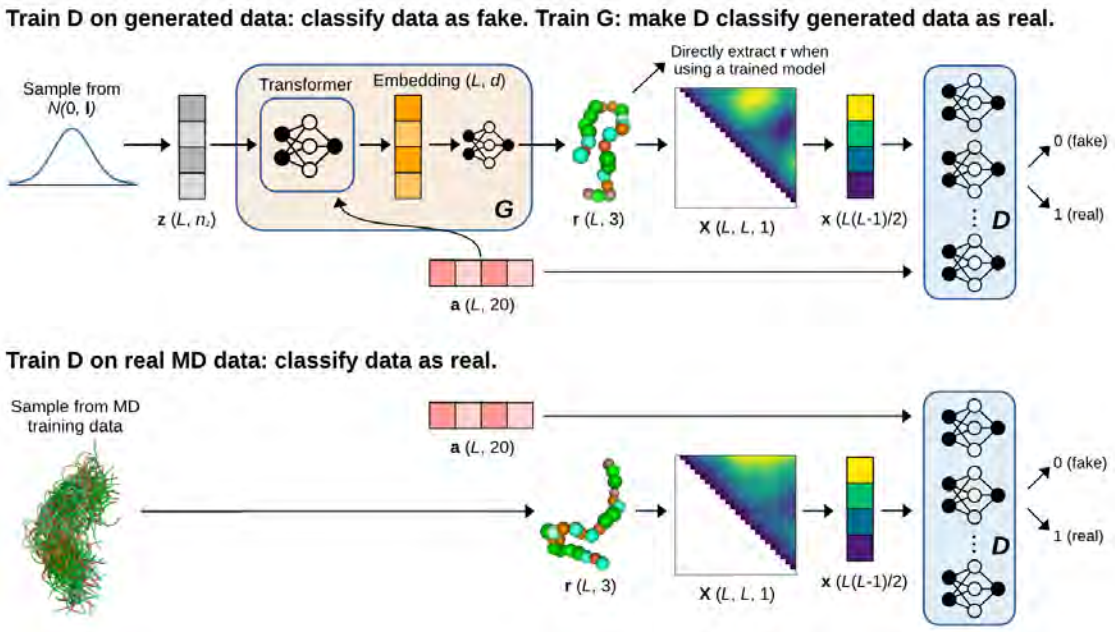

Figure 1.12: A latent sequence z and amino acid information a are inputs for the G network. The output of G is mapped to Cartesian coordinates r using a fully connected network. Conformations r generated by G and those from the training set are converted to distance matrices and, along with amino acid information a, serve as input to a set of D networks. The objective of the D network is to distinguish between real and generated samples, thereby improving the performance of G. This figure was taken from [11].

1.5 IDP Databases

Despite the continual growth of experimentally determined protein structures in resources like the Protein Data Bank (PDB) [35] and the recent emergence of the AlphaFold Protein Structure Database [36], which provides accurate structural models for millions of proteins, insights into the dynamic nature of proteins remain limited, particularly regarding the ensemble representation of IDPs.

To address this limitation, comprehensive and manually curated IDP-related databases, such as PED [37], DisProt [38], MobiDB [39], FuzDB [40], and IDEAL [41], offer multiple benefits. Primarily, they serve as foundational references and valuable resources for establishing validation pipelines to assess the reliability of IDP conformational ensembles. Additionally, these databases function as extensive training datasets for the development of future machine learning (ML) models [42].

1.5.1 The Protein Ensemble Database (PED)

The Protein Ensemble Database (PED) focuses on hosting conformational ensembles of non-globular proteins (NGPs) or regions. PED features an accessible data deposition process and upholds high standards for the quality of the structural ensembles it manages.

The PED deposition process is carried out through a web interface that enables both automatic data validation and manual curation steps. The validation step standardizes the data and enhances its quality by generating various structural indicators. The manual curation step improves data accessibility by providing detailed metadata. Biocurators use a controlled vocabulary to standardize the description of experimental methods and identify cross-references to external databases. Additionally, curators review literature to find and include ensembles not yet deposited in PED. PED also features a well-documented RESTful API for programmatic access, search, and download.

In PED, each entry is identified by the PED prefix and five digits (e.g., PED00001), representing an experiment on a protein. A PED ensemble (e.g., PED00001e001) consists of a set of conformations generated to fit the experimental data. Ensembles produced using the same proteoform (identical sequence construct and PTMs), experimental conditions, and experimental and computational methodologies are considered different replicas of the same experiment and are grouped under the same PED entry. PED also includes conformation weights as provided by the authors, indicating the probability of each conformation within the ensemble [37, 42].

1.5.2 ATLAS

ATLAS [43] is a comprehensive database of standardized all-atom MD simulations for a diverse set of representative protein structures. It comprises three distinct datasets. The main ATLAS dataset includes high-quality protein chains from the PDB. The second dataset focuses on proteins with Dual Personality Fragments (DPFs), which are regions that can exist in both disordered and ordered states across different structures. The third dataset contains MD simulations of proteins featuring chameleon sequences, which are capable of adopting different secondary structure conformations, such as α-helix or β-strand, depending on the protein context.

1.6 Analyzing IDP Ensembles

Due to the structural heterogeneity of IDPs, representing them with a single molecular structure is impossible. Instead, the entire distribution of structures, known as the structural ensemble, is used to model the different structural states. Consequently, comparing these proteins involves comparing structural ensembles rather than individual conformations. The RMSD is a common score for assessing the similarity of individual structures. However, techniques for comparing structural ensembles are limited. An initial solution was to generalize the standard RMSD method [44]. Later approaches [45] introduced three main comparison techniques based on the underlying probability distributions of the ensembles: fast harmonic algorithms for small-scale fluctuations (harmonic ensemble similarity), structural clustering methods that define similarity by the co-occurrence of conformations in both ensembles, and dimensionality reduction methods that define similarity by projecting ensembles into lower-dimensional spaces [46].

In terms of visualization, it is common to represent structural ensembles by overlaying multiple conformations in a single image. While visually appealing, such representations often fail to show the weights associated with individual conformations, thus limiting their informational content. An alternative method involves representing free energy landscapes corresponding to the structural ensembles, which can be generated using dimensionality reduction algorithms [8].

The high-dimensional nature of IDP conformational ensembles, with thousands of independently varying features, makes it difficult to uncover correlations among conformations. Efficiently identifying representative conformational substates and quantifying their populations in different contexts would greatly facilitate the identification of conformational features linked to specific functions or disease states.

One effective approach to identifying these substates in IDP ensembles is clustering. Due to its ability to provide better visualization and statistical insights, clustering is widely used in the analysis of big-data biological systems. Applications include profiling gene expression patterns, de novo structure prediction of proteins, quantitative structure–activity relationships of chemical entities, docking and binding geometry scoring, and the analysis of protein ensembles from MD trajectories. However, clustering IDP ensembles is particularly challenging due to their large conformational heterogeneity.

To address this challenge, approaches that combine dimensionality reduction techniques with clustering have proved effective. Dimensionality reduction techniques are now commonly

used in protein conformation analysis to extract low-dimensional features, with the amount of information lost depending on the dataset. For highly heterogeneous datasets like IDPs, nonlinear dimension reduction techniques are preferred as they aim to preserve the proximity of nearest neighbors, thus improving the clustering process and enabling better identification of conformational substates.

For exaple, [47] utilizes t-SNE in combination with k-means clustering to identify and visualize representative conformational substates in IDP ensembles. The recently developed t-distributed stochastic neighbor embedding (t-SNE) method effectively disentangles multiple manifolds in high-dimensional data by focusing on the local structure, thereby extracting clustered local groups of samples. As a result, t-SNE tends to outperform other methods in separating clusters and avoiding crowding.

1.7 Trajectory Analysis Software

All-atom simulations, as opposed to coarse-grained simulations, are particularly well-suited for generating sequence-specific conformational ensembles of IDPs because they allow for direct prediction of ensembles from sequence data. In response to the growing demand for all-atom simulations, various software packages have been developed to perform and analyze molecular simulations. Key packages for conducting all-atom simulations, known as simulation engines, include Amber [48], CAMPARI [49], CHARMM [50], Desmond [51], GROMACS [52], LAMMPS [53], OpenMM [54], and NAMD [55]. While most simulation engines offer some analysis routines, their usability is often limited by their design priorities, which emphasize performance.

Alongside the development of these simulation engines, there has been an increase in standalone packages designed specifically for simulation analysis. Although less robust than simulation engines, trajectory analysis software provides a lightweight and customizable alternative. These analysis tools are often developed to balance flexibility and computational efficiency. Broadly, such software can be categorized into four groups based on the knowledge required for their use and their degree of extensibility: frameworks, toolkits, plugins, and web servers (Figure 1.13).

Web servers provide access to toolkits, plugins, or frameworks through a web interface, such as a browser. These services can perform calculations like sequence alignments and trajectory analysis, and deliver results to users without the need to install external software. For molecular trajectory analysis, users simply upload a simulation trajectory and select analysis options via

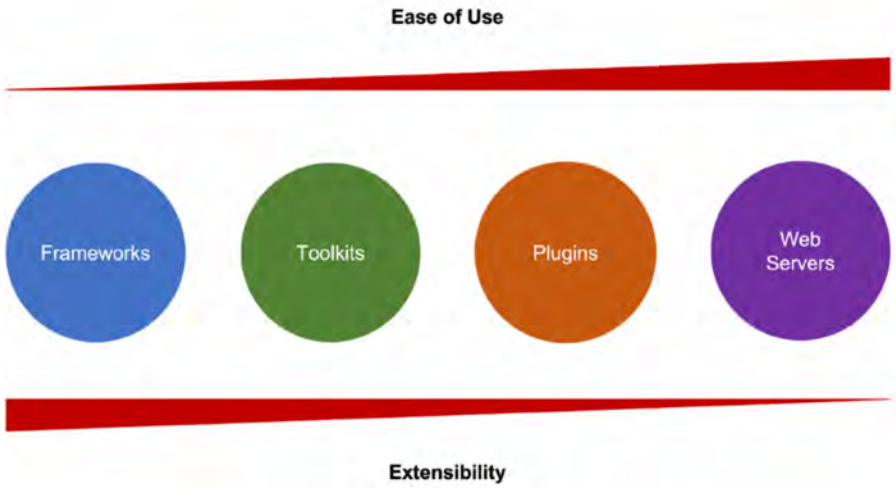

Figure 1.13: A summary of the four categories of trajectory analysis software illustrates how their ease of use and extensibility vary. As software abstractions increase, usability generally improves. However, these abstractions reduce the software's flexibility and its interoperability with other programs. This figure was taken from [12].

form controls. The server performs the analysis based on the user's specifications and reports the results back through the browser.

Plugins are programs that enhance the functionality of existing software. For trajectory analysis, plugins typically add additional analysis routines to a visualization toolkit. The primary advantage of this approach is that users can perform interactive analysis within a familiar interface without needing detailed knowledge of the host software's or plugin's implementation.

In a computational biophysical context, toolkits are collections of standalone programs or scripts designed for molecular simulations and related analyses. These tools are usually launched via a command-line interface, allowing analysis options to be directly passed as arguments to the specific program.

Frameworks provide their functionalities through an application programming interface (API), which is designed for use in a specific programming language. This API facilitates the development of new software by providing access to the framework's features and capabilities. Moreover, several frameworks organize trajectory data into natural hierarchical objects such as atoms, amino acids, proteins, protein chains, and molecules. This hierarchical organization facilitates the efficient development of customized algorithms. Consequently, these frameworks serve as ideal starting points for the creation of purpose-specific frameworks and toolkits. Examples of trajectory analysis frameworks include MDTraj [56], MDAnalysis [57, 58], LOOS [59], Jgromacs [60], OpenStructure [61], Pteros [62], CPPTraj [63], and Bio3D [64] (see Table

1.1 for details).

Software	Use	Language	Year
MDTraj	Spatial and Structural analysis	Python	2015
MDAnalysis	Spatial and Structural analysis	Python	2011
LOOS	Spatial, Structural, and information Theoretic Analyses	C++/Python	2009
Jgromacs	Object-Oriented Representation and Analysis	Java	2012
OpenStructure	Spatial, Structural, and information Theoretic Analyses, with Visualization	C++/Python	2013
Pteros	Geometry transformations, Structural alignment, Energy calculations	C++	2012
CPPTRAJ	Spatial, Structural, and information Theoretic Analyses	C++	2013
Bio3D	Spatial, Structural, Evolutionary Analysis, with Visualization	R	2006

Table 1.1: Frameworks for Analysis of MD Trajectories

1.7.1 MDTraj

MDTraj is an open-source Python library designed for the analysis and manipulation of MD trajectories. It can load trajectory and topology data from a variety of MD package formats, offering wide interoperability. By leveraging Python and NumPy, MDTraj connects MD data with an extensive ecosystem of data science tools. Rather than providing all functionalities within a single toolkit, MDTraj integrates MD data with a broad range of statistical and graphical libraries, such as scikit-learn and matplotlib. Additionally, MDTraj features an atom selection language, enabling users to apply analysis functions to specific subsets of atoms within the system.

MDTraj simplifies data analysis for MD by providing efficient tools to extract order parameters and define distance metrics between simulation snapshots. It features a highly optimized RMSD engine, performing Theobald's QCP algorithm three times faster than the original. Additionally, it offers functions for secondary-structure assignment, solvent-accessible surface area determination, hydrogen bond identification, residue-residue contact mapping, NMR scalar coupling constants, nematic order parameters, and extraction of internal degrees of free-

dom. These functions, optimized with C/C++ and multithreading, return data as multidimensional NumPy arrays, ensuring compatibility with the scientific Python ecosystem.

Additionally, MDTraj includes a unique interactive WebGL-based three-dimensional structure viewer designed for the IPython notebook, adapted from iview. This viewer is complemented by MDTraj's TrajectoryView widget, which operates directly within the IPython notebook to deliver a high-quality, fully interactive three-dimensional rendering of molecular trajectories. Users can also save their visualizations as high-quality PNG images or STL 3D models.

MDTraj is utilized in various toolkits for analysis and input processing. Examples include SSTMap [65], taurenmd [66], and TTClust [67] toolkits. Additionally, frameworks such as Geo-Measures [68], MDEntropy [69], PyRETIS [70], OpenPathSampling [71], mdciao [72], ProLint [73], and the Biotite [74] package integrate MDTraj into their APIs for diverse functionalities (see Table 1.2 for details). Notably, SOURSOP [12] stands out as a Python-based software package dedicated specifically to analyzing conformational ensembles of IDPs, relying on MDTraj for its core functionality. SOURSOP integrates analysis routines typically applied to folded proteins with a suite of extensively utilized IDR-centric analyses.

Software	**Use**	**Year**
SSTMap	Thermodynamic, Structural, and Spatial Analysis of Water in MD trajectories	2018
taurenmd	Spatial and Structural analysis	2020
TTClust	Identify dissimilar conformations through trajectory clustering	2018
Geo-Measures	PyMol plugin for structure ensemble analysis	2020
MDEntropy	Information Theoretic analysis	2017
PyRETIS	Identifies rare events by replica exchange	2017
OpenPathSampling	Transition Path Sampling	2019
mdciao	Structural analysis and visualization using residue-residue distances	2022
ProLint	Analysis and visualization for lipid-protein interactions	2021
Biotite	Spatial and Structural analysis	2018
SOURSOP	Integrates analysis routines typically applied to folded proteins with IDR-centric analyses.	2023

Table 1.2: Trajectory Analysis Software Built upon MDTraj

1.8 Thesis Objectives

For intrinsically disordered proteins (IDPs), it is common to obtain several different structural ensembles of the same protein using various methods or under different experimental conditions. Despite the abundance of software available for analyzing protein structures and several solutions tailored for analyzing structural ensembles of IDPs, there is a notable shortage of tools that provide rapid analysis routines for comparing these ensembles.

The goal of this thesis is to showcase the development of a software package specifically designed for analyzing and comparing Structural Ensembles of IDPs. As these structural ensembles are represented by molecular trajectories, the final product can be categorized as trajectory analysis software. The main objectives are to create a lightweight, user-friendly, and easily installable software solution that is both extendable and integrated with existing tools. Additionally, the software aims to analyze multiple IDP ensembles within a single data analysis pipeline, offering a variety of analysis routines that include methods from polymer physics to dimensionality reduction techniques.

To meet these requirements, IDPET (Intrinsically Disordered Protein Ensemble Toolkit) was developed. IDPET is a Python package designed for the comprehensive analysis of conformational ensembles of IDPs. Built on the MDTraj library, IDPET can read ensemble files in various formats and automatically download data from databases such as the Protein Ensemble Database (PED) and ATLAS.

IDPET offers a range of functionalities for analyzing and comparing multiple structural ensembles of disordered proteins in parallel. It includes various visualization tools that enable the exploration of global and local features of IDPs, offering insights into their general configurations and detailed interactions at various scales. These visualizations are crucial for understanding the complex structural dynamics inherent to IDPs.

Moreover, IDPET incorporates advanced statistical methods and dimensionality reduction techniques to manage the complex, high-dimensional nature of IDP conformational ensembles. By combining dimensionality reduction with clustering, IDPET effectively identifies and visualizes representative conformational substates within IDP ensembles.

2 Methods

2.1 Software Architecture and Design of IDPET

The Intrinsically Disordered Protein Ensemble Toolkit (IDPET) is developed in Python 3.10 and relies heavily on the simulation analysis framework MDTraj for many of its functionalities. Firstly, IDPET utilizes MDTraj for parsing and generating molecular dynamics trajectories. These trajectories, represented as mdtraj.trajectory objects (with access to mdtraj.topology objects), serve as the backbone for many of the analyses provided by IDPET. Following Python conventions already enforced through MDTraj, IDPET aims to represent all data obtained from these trajectories in the NumPy array format whenever possible, promoting extensibility and interoperability with other tools.

One of the main goals of this package is to be lightweight, easy to install, and user-friendly. To achieve this, IDPET follows a straightforward file structure, promoting the use of lower-level features for experienced users (Figure 2.1). The files contain analysis routines at different functional levels, intuitively distributed across various modules. While the main functionalities of the package can be found within the ensemble_analysis and visualization modules, dimensionality_reduction and ensemble modules possess user-friendly interfaces that makes it possible to utilize their functionalities aside from the main pipeline. The featurization directory offers methods for feature extraction, categorized into four modules based on their semantics. The angles and distances modules provide methods for extracting local (residue-level) features, the

glob module focuses on global (conformation-level) features, and the ensemble_level module provides methods for computing ensemble-level metrics. IDPET is available for installation via pip and conda tools and features extensive documentation. Additionally, several Jupyter notebooks have been developed to demonstrate the various use cases of the package.

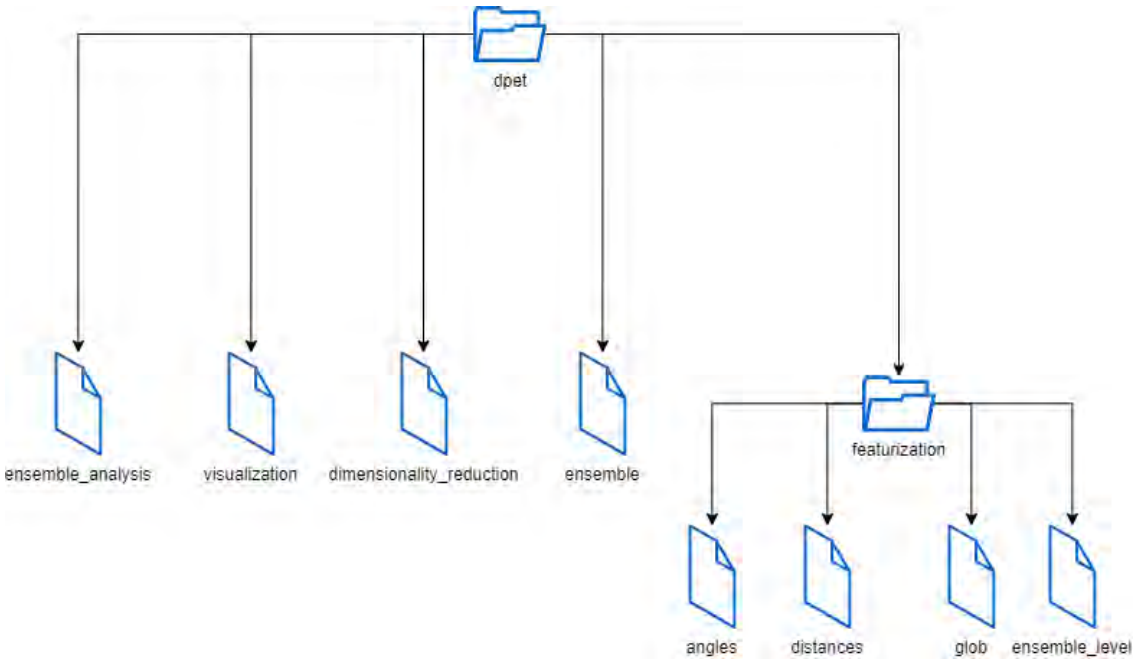

Figure 2.1: The straightforward structure of the programmatic library ensures ease of use while promoting the utilization of lower-level functional features. Only the modules with relevant analysis routines are displayed.

Furthermore, IDPET follows an object-oriented design structure to encapsulate its functionalities and provide an intuitive interface. While leveraging the multi-paradigmatic nature of Python, IDPET predominantly uses the object-oriented paradigm to distribute its functionalities across a few key objects. This design choice, inspired by similar frameworks such as MDTraj and SOURSOP, aims to enhance ease of use. Users can access all the tools provided by IDPET by familiarizing themselves with these core objects. A class diagram showcasing the object-oriented design of IDPET is shown in Figure 2.2.

The two main objects in the IDPET framework are EnsembleAnalysis and Visualization, which offer high-level functionalities sufficient for basic analysis of IDPs. The EnsembleAnalysis object contains the data transformation pipeline that begins with data loading and extends to feature extraction, dimensionality reduction, and clustering methods. Although EnsembleAnalysis houses all the products of the various pipeline steps, this data is readily accessible in the form of NumPy arrays or dictionaries with ensemble IDs as keys and arrays as values. The Visualization

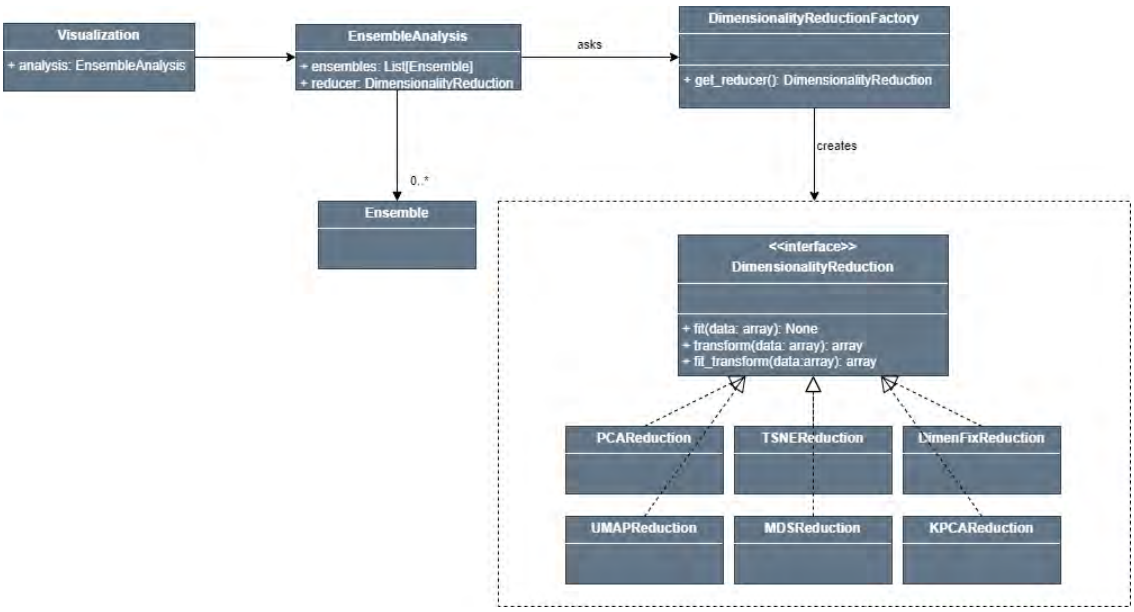

Figure 2.2: A UML class diagram depicting the main classes present in IDPET. The high-level classes EnsembleAnalysis and Visualization interact with lower-level ones such as Ensemble and the classes implementing the DimensionalityReduction interface.

class builds on this pipeline, initialized with the EnsembleAnalysis object after data transformation is complete, to access all produced data. It provides numerous plotting functions that offer insights into the analyzed IDPs.

IDPET is designed for the simultaneous analysis and comparison of multiple ensembles, requiring a dedicated unit to represent each ensemble individually. The lower-level Ensemble object contains ensemble-specific data and includes separate pipeline steps applied independently. At a higher level, the EnsembleAnalysis object is instantiated with a list of Ensemble objects, specifying their data sources and processing instructions, and oversees these objects throughout the analysis process.

In IDPET, dimensionality reduction methods are managed through the Factory design pattern, specifically the Factory Method pattern, which is a common software engineering approach. This pattern offers an interface that allows a client to obtain instances of classes that adhere to a specific interface or protocol, without needing to know the exact class being instantiated. This approach enhances encapsulation and code reuse, as implementations can be altered without requiring any modifications to the client code.

The factory pattern offers several benefits. Firstly, it enforces the dependency inversion principle, ensuring that client dependencies are only on abstract classes and interfaces, not on the

concrete subclasses they receive. Secondly, it decouples the concrete product instances from all aspects except their point of instantiation. Thirdly, the factory pattern promotes the creation of consistent products, as they are all instantiated using the same factory [75].

The various dimensionality reduction classes, each implementing the abstract class DimensionalityReduction, are instantiated through the static methods of the DimensionalityReductionFactory class. This design promotes modularity by enabling easy addition and integration of new methods. Although all dimensionality reduction methods adhere to a common interface that includes methods like fit, transform, and fit_transform, their implementations can differ considerably. Additionally, some methods integrate clustering techniques into their pipelines, each with its unique approach. Because the various methods adhere to the same interface defined in DimensionalityReduction, the different parameters for each method are passed directly to the constructors of the subclasses upon creation, using named and positional arguments. In the example given in Listing 2.1, the reference to the hidden TSNEReduction subclass is obtained using the DimensionalityReductionFactory class, which is then used through the uniform interface defined in DimensionalityReduction to perform the fit_transform method.

```
1
2 # Obtain a dimensionality reduction subclass using the factory with all
      parameters provided
3 reducer = DimensionalityReductionFactory.get_reducer(
4     method='tsne',
5     perplexity_vals=[10, 20, 50, 100],
6     circular=False,
7     range_n_clusters=range(2, 10, 1)
8 )
9
10 # Apply the dimensionality reduction to the data
11 reduce_dim_data = reducer.fit_transform(data)
```

Listing 2.1: Using the Factory Pattern for Dimensionality Reduction

IDPET uses SPHINX for documentation and Git for version controll. Additionally, IDPET relies on NumPy for numerical computations on arrays and matrices, scikit-learn for machine learning, SciPy for scientific and technical computing required in certain analysis routines, Requests for making HTTP requests to remote databases, matplotlib for visualizations and provides outputs as Pandas dataframes.

2.2 IDPET Workflow

In this section, the workflow of IDPET is outlined. The IDPET analysis follows a roughly sequential structure, where each step builds upon the previous one, as illustrated in Figure 2.3. While different steps of the pipeline may follow different paths, they can generally be divided into three main stages: data loading, analysis, and visualization. The corresponding modules of the software package responsible for these stages are the ensemble, ensemble_analysis, and visualization modules, respectively.

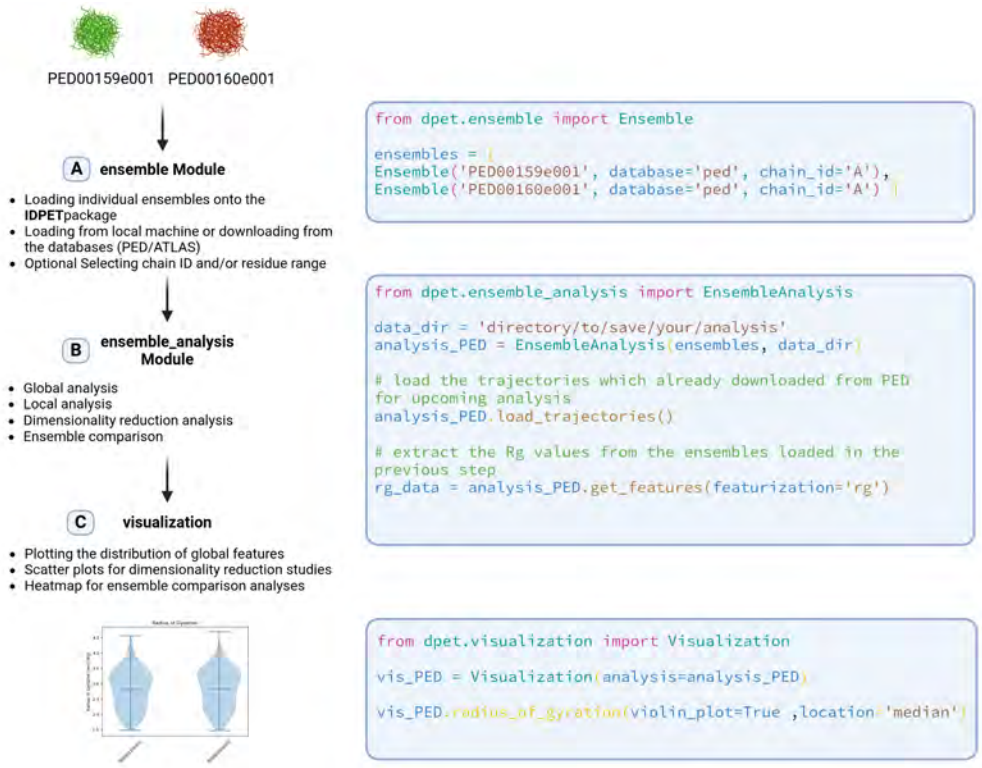

Figure 2.3: A schematic of the IDPET workflow is shown, starting with data loading from local files or IDP databases using the ensemble module. It proceeds with four types of analyses: global, local, dimensionality reduction, and ensemble comparison, performed by the ensemble_analysis module. Finally, different visualization options are provided by the visualization module.

The workflow commences with data loading from either the local file system or a supported IDP database. MDTraj is employed during this initial phase to parse or generate trajectory objects from the provided data. Following data loading, the analysis proceeds with various functions categorized into global analysis, local analysis, dimensionality reduction analysis, and

ensemble comparison. Global analysis focuses on entire conformations within the analyzed ensembles, while local analysis provides residue-level information. Dimensionality reduction, combined with clustering techniques, condenses the data into a two-dimensional space, facilitating visualization and the identification of structural patterns. Finally, ensemble comparison methods quantify relationships between entire ensembles using a set of similarity measures. These functionalities leverage scikit-learn and other Python machine learning libraries. An additional option is feature extraction, which is integrated into the other analyses but can also be used independently by the user to obtain global or local features of the ensembles for separate evaluation. IDPET supports its analysis routines with a variety of visualization functions powered by matplotlib. Most visualization functions are designed to be customizable and extendable.

2.3 Data Loading

As previously mentioned, IDPET utilizes MDTraj to generate trajectories from various input files or to load them directly from specific trajectory files. Currently, IDPET supports data files in PDB format, directories containing PDB files (each representing a different conformation"), and trajectory files in "XTC" or "DCD" formats along with a corresponding topology file in PDB format. These files can be loaded from the local machine. Additionally, IDPET is integrated with the data APIs of two protein structure databases, PED and ATLAS, enabling it to fetch necessary data by providing the correct database ID. A summary of available data formats is presented in Table 2.1

The `EnsembleAnalysis` object is instantiated by providing a list of `Ensemble` objects and a path to an output directory where the various files produced during the analysis pipeline are stored. Each `Ensemble` object must contain a unique code serving as its main identifier throughout the analysis, along with paths to the data files or an identifier from one of the supported databases (`"ped"` or `"atlas"`). Additional optional parameters include a chain ID, used to select a single chain if a provided pdb file contains multiple chains, and a residue range, which indicates the indexes of residues to be analyzed (See Listing 2.2 for an example of the data loading step).

Since data loading can be a relatively long task, measures were taken to skip certain steps wherever possible. For instance, downloading ensemble data from remote databases is skipped when the processed data is already present in the output directory. Additionally, when trajectories are generated from data files using MDTraj, the trajectory files are immediately saved to bypass this lengthy step in future analyses.

Data Format	Short Description	Technical Details
PDB Files	Single PDB files representing protein structures	Standard file format for macromolecular structure data; contains atomic coordinates, connectivity, and optionally, occupancy and temperature factors
Directories with Multiple PDB Files	Collections of PDB files in a directory	Directory-based input for batch processing of structural ensembles, where each PDB file represents a distinct conformational state
Trajectory Files	Trajectories (xtc or dcd) along with a topology file (pdb)	XTC and DCD are binary trajectory formats capturing atomic positions over simulation time; requires a corresponding PDB file for atomic topology and connectivity information

Table 2.1: Supported File Formats for IDPET Analysis

An important step in data loading is determining whether a loaded trajectory belongs to a coarse-grained model. This is identified by checking if the topology contains only C atoms. If so, the ensemble is marked as coarse-grained and is treated differently in subsequent pipeline steps. In fact, some features are completely unavailable if at least one ensemble comes from a coarse-grained model.

Since most functionalities provided by IDPET are designed to work on a single polypeptide chain, conformations consisting of multiple chains need special handling. By convention, different chains are assigned alphabetical identifiers (A, B, C, etc.). However, MDTraj trajectory objects do not retain these identifiers, instead replacing them with numerical indexes. To address this, IDPET processes the original PDB files to list all chain identifiers and map them to indexes. This allows users to set the `chain_id` parameter of the `Ensemble` objects before loading the ensembles, automatically restricting the analysis to the specified chain.

```
1
2 # Define a list of Ensemble objects
3 ensembles = [
4     Ensemble(ens_code='Ensemble1', database='atlas'), # Ensemble1 from the
          'atlas' database
5     Ensemble(ens_code='Ensemble2', database='ped'), # Ensemble2 from the '
          ped' database
6     Ensemble(ens_code='Ensemble3', data_path='/path/to/local/file1.pdb'), #
           Ensemble3 from a local PDB file
7     Ensemble(ens_code='Ensemble4', data_path='/path/to/local/file2.dcd',
          top_path='/path/to/local/file2.top.pdb') # Ensemble4 from a local
          trajectory file with topology
8 ]
9
10 # Instantiate the EnsembleAnalysis object
11 analysis = EnsembleAnalysis(ensembles=ensembles, output_dir='/path/to/
       output/directory')
12
13 # Load trajectories for all ensembles
14 analysis.load_trajectories()
15
16 # Randomly sample 50 conformations from the ensembles
17 analysis.random_sample_trajectories(sample_size=50)
```

Listing 2.2: Data loading example. The EnsembleAnalysis class is instantiated with a list of Ensembles. For each ensemble it is necessary to specify local paths to data files or indicate the remote database. After loading the ensembles it is possible to randomly sample a given number of conformations

A function for randomly sampling a defined number of conformations from all loaded ensembles is supported. When this function is performed, the original trajectory representation of the ensembles is stored, allowing the functionality to be executed multiple times, each time overriding the previous sampling. Interestingly, this implementation corresponds to randomly sampling frames from molecular dynamics trajectories. However, in the context of IDPET, different conformations belonging to an ensemble are treated as simulation frames.

2.4 Global Analysis

Using this toolkit, multiple structural ensembles of intrinsically disordered proteins (IDPs) can be analyzed simultaneously. A structural ensemble contains several conformations, which rep-

resent three-dimensional protein structures, along with their statistical weights, indicating their prevalence under experimental conditions. The Global Analysis component of the pipeline offers functionalities to extract information about whole conformations, rather than focusing on residue-level details.

IDPET offers several functionalities to gain insights into the global features of conformational ensembles. In addition to extracting these features and presenting them to the user, the global analysis functionalities are closely integrated with visualization functions. Consequently, various distribution functions plot the values of these global features across entire ensembles, as each value corresponds to a conformation. Additionally, several routines connect these features, enabling the plotting of one feature against another for comparative analyses. IDPET's global analysis evaluates features such as the radius of gyration, end-to-end distances, asphericity, prolateness, global surface accessible solvent area (SASA), and the Flory scaling exponent.

2.4.1 Radius of Gyration

The radius of gyration R_g is a measure of the compactness of the entire protein structure and is computed in two steps. First, the coordinates of the center of mass R_C are determined, disregarding the hydrogen atoms, using the following equation:

$$\sum m_i(\mathbf{r}_i - \mathbf{R}_C) = 0,$$

where m_i is the mass of the i-th atom and $\mathbf{r}_i$ denotes its coordinates.

From this, the radius of gyration is obtained as:

$$R_g^2 \approx \frac{1}{N}\sum_{i=1}^{N}(\mathbf{r}_i - \mathbf{R}_C)^2,$$

where N is the number of atoms other than hydrogens in a protein [76].

A lower radius of gyration indicates a more compact protein structure, where atoms are tightly clustered around the center of mass, typically corresponding to a folded and functional state. Conversely, a higher radius of gyration indicates a less compact, more extended structure, often seen in unfolded or intrinsically disordered proteins.

The radius of gyration is a measurable physical quantity that can be both experimentally determined and calculated without a reference structure. Plotting values against the radius of gyration is commonly used to study protein structure stability [77].

2.4.2 End-to-end Distances

The end-to-end distance in a polypeptide chain is the distance between the *Cα* atoms of its N- and C-termini. This measurement is significant as it reflects the overall span of the chain and can be experimentally determined without needing a reference structure. Like the radius of gyration, the end-to-end distance provides insight into the compactness of a protein structure. Therefore, it serves as a useful metric for comparing and correlating with other structural measures of proteins [77].

2.4.3 Asphericity

While Rg provides a good measure for the size of an IDP, the asphericity provides a very useful measure of its shape [78]. In order to compute the asphericity of a protein, it is first necessary to compute the gyration tensor.

The gyration tensor $S_{\alpha\beta}$ is a 3x3 matrix that provides information about the distribution of a molecule's mass around its center of mass. It is defined as:

$$S_{\alpha\beta} = \frac{1}{N}\sum_{i=1}^{N}(r_i - R_{\mathrm{C}})_\alpha (r_i - R_{\mathrm{C}})_\beta$$

where N is the number of atoms in the molecule, r_i is the position vector of the i-th atom, R_{C} is the position vector of the center of mass of the molecule, α and β denote Cartesian coordinates (x, y, z).

To gain insight into the molecule's shape, the eigenvalues $(\lambda_1, \lambda_2, \lambda_3)$ of the gyration tensor are computed. These eigenvalues represent the principal components of the tensor and provide information about the distribution of the molecule's mass along the principal axes.

In this context, the radius of gyration R_g is a measure of the overall size of the molecule and is related to the eigenvalues of the gyration tensor:

$$R_g^2 = \lambda_1 + \lambda_2 + \lambda_3$$

Asphericity is a measure of how much the shape of the molecule deviates from being spherical. It is defined using the eigenvalues of the gyration tensor as follows:

$$A = 1 - \frac{3(\lambda_1\lambda_2 + \lambda_1\lambda_3 + \lambda_2\lambda_3)}{(\lambda_1 + \lambda_2 + \lambda_3)^2}$$

The value of A ranges from 0 to 1, with 0 representing a sphere and 1 representing a thin rod [79, 80].

2.4.4 Prolateness

In addition to metrics like the radius of gyration and asphericity, the eigenvalues of a protein's gyration tensor provide insights into its shape, specifically its prolateness. Prolateness measures how elongated a structure is relative to its transverse dimensions. A prolateness value greater than one indicates a more elongated shape, whereas smaller values suggest a more spherical shape.

The prolateness (*Prolateness*) of a protein is computed using the following formula:

$$Prolateness = \frac{\lambda_{mid} - \lambda_{min}}{\lambda_{max}}$$

where λ_{max}, λ_{mid}, and λ_{min} represent the largest, middle, and smallest eigenvalues obtained from the gyration tensor, respectively.

2.4.5 Surface Accessible Solvent Area (SASA)

An important characteristic of each residue in a protein structure is its solvent accessible surface area (SASA), which quantifies the portion of the residue's surface exposed to a solvent, such as water, enabling potential interactions with other proteins or smaller molecules.

SASA was first described by Lee and Richards in 1971 [81] and is sometimes referred to as the Lee-Richards molecular surface. SASA is typically calculated using the "rolling ball" algorithm developed by Shrake and Rupley in 1973 [82]. This method involves simulating the rolling of a probe ball over the protein surface while considering the van der Waals radii of the atoms (Figure 2.4).

The core concept of this algorithm is to generate a mesh of points representing the surface of each atom, positioned at a distance equal to the sum of the atom's van der Waals radius and the probe radius. The algorithm then counts the number of these mesh points that are on the molecular surface, meaning they are not within the radius of any other atom.

From a biological perspective, accessibility is significant because it accounts for residues buried deep within the folded protein structure which greatly impact the protein's stability, even though they may not directly interact with other molecules [2].

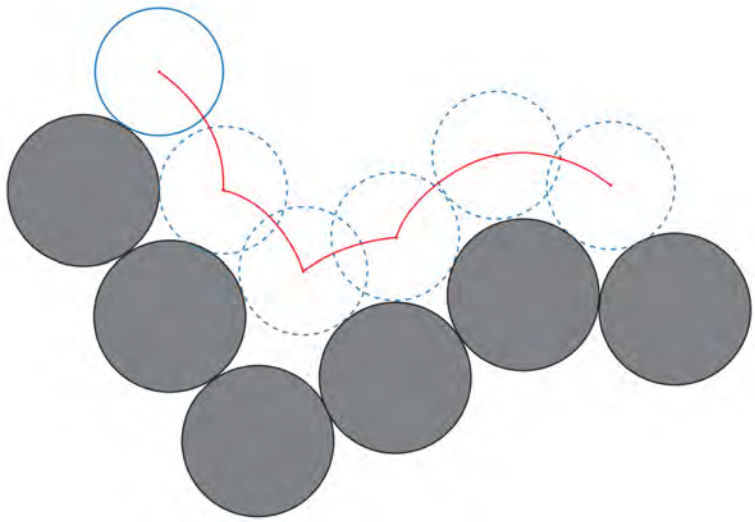

Figure 2.4: Solvent accessible surface area (SASA) refers to the portion of a protein's surface that is in direct contact with the solvent, typically water. SASA is determined by rolling a probe, represented as a blue sphere the size of a water molecule, over the protein's surface. The protein's surface is defined by the van der Waals volumes of the amino acid atoms, shown in gray. The SASA is then described by the path of the probe's center, which moves along the surface, shown in red. This figure was taken from [2].

2.4.6 Flory Scaling Exponent

A polymer chain, such as a polypeptide, behaves differently in various solvents. In a good solvent, where the polymer is highly soluble, the polymer chain expands to maximize its contacts with the solvent. Conversely, in a poor solvent, where the polymer is insoluble or barely soluble, the polymer chain condenses to form a spherical structure. Scaling laws relate the radius of gyration of a polymer (representing its size) to its length under different solvent conditions:

$$R_g \propto N^{\nu}$$

where R_g represents the radius of gyration of the polymer, N is the number of bonds, and ν is the Flory scaling exponent.

When $\nu = \frac{3}{5}$, the chain is in the expanded coil state. When $\nu = \frac{1}{3}$, it indicates the most compact globule state. When $\nu = 0.5$, the polymer is in the θ state, behaving like a perfect chain [83].

For the purposes of IDPET, Flory scaling exponents (ν) were determined by fitting the mean-squared distances between residues ($R_{\langle ij \rangle^2}$), calculated for sequential separations greater than five residues along the linear sequence. This method is detailed in [84].

2.5 Local Analysis

IDPET's local analysis focuses on residue-level features of proteins that define their three-dimensional structure, closely integrating with visualization tools. Typical plots of local features display the residue index on the X-axis and the examined local feature on the Y-axis. Key visualizations include Ramachandran plots for torsion angles, average distance maps, and contact probability maps for *Cα-Cα* distances. Local analysis routines encompass these visualizations along with α angle distributions, secondary structure propensities, and site-specific order and flexibility parameters.

2.5.1 Average Distance and Contact Probability Maps

Spatial proximities between amino acid residues are crucial for protein 3D structure prediction. This information can be represented as a square symmetric matrix, where values indicate Euclidean distances between specific atoms, known as a distance map. When the matrix contains binary data about residue-residue interactions, qualified by a Euclidean distance below a set threshold, it is called a contact map. Most approaches utilizing contact maps for protein structure prediction use Euclidean distances between *Cα* or *Cβ* atoms, with a contact threshold typically set at 8 Å [85].

Since IDPET analyzes ensembles containing multiple protein structures, its distance maps require averaging across all conformations. The average distance map in IDPET illustrates the average distance between every pair of *Cα* atoms across all conformations within an ensemble.

Similarly, contact maps in IDPET are adjusted to encompass information from multiple structures. IDPET computes contact maps for each conformation, identifying pairs of *Cα* atoms that are closer than a predefined threshold (typically 8 Å). These individual contact maps are then averaged to derive a contact probability for every residue pair in the ensemble. The resulting probabilities are visualized as a contact probability map.

2.5.2 Ramachandran Plots

Dihedral angles, also known as torsion angles, are particularly useful for compactly representing the backbone or overall topology of protein structures. The two key backbone dihedral angles are phi (ϕ), along the N-Cα bond, and psi (ψ), along the Cα-C bond of the residue. Each residue in a protein is characterized by a ϕ angle and a ψ angle. Consequently, the complete protein backbone can be described by the collection of ϕ and ψ angles for all residues in the protein.

When plotting torsion angles of known protein structures, with ϕ on the X-axis and ψ on the Y-axis, a distinct distribution known as a Ramachandran plot emerges (Figure 2.5). These plots reveal empty regions, comprising about 75% of the area, where clashes between amino acid side chains restrict access to certain torsion angles. Furthermore, residues with torsion angles in the upper left quadrant (centered around ϕ = -120° and ψ = 120°) typically correspond to beta strands, while those in the lower left quadrant (centered around ϕ = -60° and ψ = -40°) correspond to alpha helices. Ramachandran plots are invaluable for evaluating the quality of protein structures. Studies have shown that high-quality structures exhibit tightly clustered patterns in these plots, with few residues falling outside the defined "allowed" dihedral regions [2].

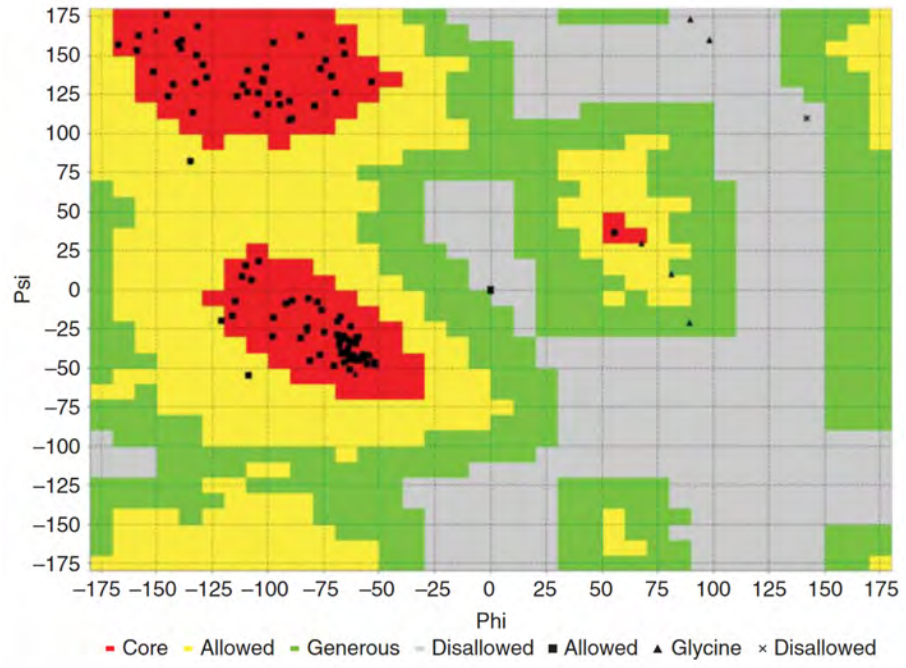

Figure 2.5: A Ramachandran plot for the thioredoxin protein (PDB ID: 2TRX) displays each residue as a point, categorized by specific symbols. Squares represent residues within the "allowed" or "core" regions, triangles denote glycine residues, and "X" marks residues in "disallowed" regions. Red areas mark "core boundaries," where about 85% of residues in high-quality structures should be found. Yellow areas denote "allowed boundaries," where around 10% of residues should lie. Residues falling within the "generously allowed boundaries" (green regions) or outside these areas suggest potential steric issues. Glycine residues, uniquely marked with "X," can appear across the entire plot. This figure was taken from [2].

IDPET utilizes MDTraj to computeφ and ψ angles for a trajectory to construct Ramachandran plots for the analyzed ensembles. There are two options for a Ramachandran plot. The first option plots a two dimensional histogram with a user-defined number of bins. The second

option is more straightforward as it simply scatters values of all ensembles onto a single plot.

2.5.3 Cα-based Torsion Angles (α Angles)

Cα-based torsion angles as defined in [86], referred to as α angles, are a convenient tool that can easily and without bias identify most regions of the protein where conformational changes take place. Given a set of coordinates for a protein, a dihedral angle α(i) for each residue i is defined by the atoms Cα(i - 1), Cα(i), Cα(i + 1), and Cα(i + 2). First, a vector P is calculated, which is normal to the plane formed by the vectors Cα(i - 1)-Cα(i) and Cα(i + 1)-Cα(i). Similarly, a vector Q is calculated, which is normal to the plane formed by the vectors Cα(i)-Cα(i + 1) and Cα(i + 2)-Cα(i + 1). The angle between vectors P and Q determines the value of the dihedral angle α(i). Two protein structures can be compared by constructing their Δα profiles, which depict the differences in alpha angles determined for corresponding residues. These Δα values exceeding a predefined threshold can identify regions where the directions of the polypeptide chains differ significantly.

In IDPET, alpha angles are calculated using MDTraj to determine dihedrals defined by sets of four consecutive C alpha atoms within a trajectory. These angles represent torsional movements around bonds linking the middle two atoms, as defined by planes formed by the first three and last three atoms. Analysis involves creating histograms to visualize the frequency or density of these alpha angles within distinct ensembles.

2.5.4 Secondary Structure Propensities

For a successful analysis of the relationship between amino acid sequence and protein structure, it is crucial to have a clear and physically meaningful definition of secondary structure. This can be achieved through a set of simple, physically motivated criteria based on pattern recognition of hydrogen-bonded and geometrical features.

Characterizing secondary structures using backbone φ, ψ angles or C_α positions requires adjusting multiple parameters, such as four angles for a rectangle in the φ, ψ plane for each type of secondary structure. In contrast, the presence or absence of a hydrogen bond can be determined by a single decision parameter: a cutoff in bond energy. Therefore, secondary structures can be recognized primarily by identifying hydrogen bonding patterns. These patterns include "n-turns," characterized by a hydrogen bond between the CO group of residue i and the NH group of residue $i + n$ (where $n = 3, 4, 5$), and "bridges," which involve hydrogen bonds between residues that are not sequentially adjacent. These patterns encompass nearly

all backbone-backbone hydrogen bonds. Repeating 4-turns define α-helices, and repeating bridges define β-structures. Additional patterns include 3_{10}-helices, π-helices, single turns, and single β-bridges [87].

The Dictionary of Protein Secondary Structure (DSSP) is commonly used to describe protein secondary structures with single-letter codes based on hydrogen bonding patterns. DSSP defines eight types of secondary structures: 3-turn helices (G), 4-turn helices (H), 5-turn helices (I), hydrogen-bonded turns (T), extended strands in β-sheet conformations (E), residues in isolated β-bridges (B), bends (S), and coils (C).

IDPET relies on MDTraj's implementation of the DSSP algorithm to perform local analysis of ensemble conformations. Because IDP ensembles contain many conformational states, the relative content of specific structures is considered. The analysis takes as a parameter the code for the secondary structure and plots the relative presence of that structure at specific residues across all conformations.

2.5.5 Site-specific Order

A site-specific measure of order can be derived from the distribution of angles (θ_{ij}) between bond vectors, where each bond corresponds to the $C\alpha_i - C\alpha_{i+1}$ vector. At any given residue i, the variance σ^2_{ij} of cos θij is computed across all residues j, including i. The correlation parameter is defined as $c_{ij} = 1 - \sqrt{2}\sigma^2_{ij}$. The local order parameter for residue i is the average of all c_{ij} values:

$$o_i = \frac{1}{N}\sum_{i=1}^{N} c_{ij}$$

This parameter ranges between 0 for a random distribution of all θ_{ij} angles and 1 for perfect order [88, 89].

2.5.6 Site-specific Flexibility

To obtain a disorder parameter that reflects local flexibility, the circular variance of the Ramachandran angles φ_i and ψ_i of all conformers is considered. The circular variance of φ_i is given by:

$$R^2(\varphi_i) = \left(\sum_{c=1}^{C} w_c \cos(\varphi_{i,c})\right)^2 + \left(\sum_{c=1}^{C} w_c \sin(\varphi_{i,c})\right)^2$$

where C is the number of conformers in the ensemble, w_c is the weight associated with the c-th conformer, and $\varphi_{i,c}$ is the φ_i angle belonging to the the c-th conformer. An analogous expression applies for $R^2(\psi_i)$.

If all conformers have the same dihedral angles at residue i, each of the two circular variances is equal to one. For a large C and with a uniform distribution of dihedral angles, the circular variance tends to zero. Therefore, a site-specific flexibility parameter can be defined as follows:

$$f_i = 1 - \frac{1}{2}R(\varphi_i) - \frac{1}{2}R(\psi_i)$$

This parameter ranges from 0 when all conformers at residue i have identical dihedral angles, to 1 when there is a uniform distribution of dihedral angles at residue i [88, 89].

2.6 Feature Extraction

IDPET supports various methods for extracting structural and biophysical features from the analyzed ensembles. These features can be utilized independently by the user, and some can be further employed in specific machine learning dimensionality reduction techniques.

Dimensionality reduction algorithms can be applied only to certain local features of the ensemble. These features include *Cα-Cα* distances, φ and ψ torsion angles, α angles and φ and ω interresidue orientation angles as defined in the trRosetta method [90]. *Cα-Cα* distances refer to the Euclidean distance between each pair of residues in a conformation, φ and ω are angles defined along the *N-Cα* bonds and the *Cα-C* bonds of each residue of the protein, and α angles are torsion angles defined by four consecutive *Cα* atoms around each residue.

Usually, ω and φ angles denote the third and first mainchain backbone dihedrals of α amino acids, respectively. Meanwhile, theta refers to the second mainchain backbone dihedral of β amino acids. However, in the context of trRosetta, ω signifies the dihedral angle between two residues defined between the *Cα* and *Cβ* atoms of the first residue and the *Cβ* and *Cα* atoms of the second residue. Theta refers to the dihedral angle between two residues constructed by the *N*, *Cα*, and *Cβ* atoms of the first residue and the *Cβ* atom of the second residue. φ refers to the dihedral angle between two residues that lies between the *Cα* and *Cβ* atoms of the first residue and the *Cβ* atom of the second residue.

In designing IDPET, the goal was to enable the entire analysis pipeline to be managed by a single object. The resulting EnsembleAnalysis object, initialized with ensemble data and used to load molecular dynamics trajectories, also extracts features and performs dimensionality reduc-

tion on those features. The primary function for this task, extract_features, requires only the feature extraction method as a mandatory parameter, with other method-specific parameters being optional. This function returns a dictionary with ensemble IDs as keys and the extracted features as values, while also updating the state of the EnsembleAnalysis object for further processing. The extracted features are stored within the EnsembleAnalysis class and can be accessed at any time.

In addition to the feature extraction that is linked with subsequent dimensionality reduction methods, IDPET also supports the extraction of all other local and global features of an ensemble, which were described in the previous sections. The function that serves this purpose, get_features, operates similarly but does not alter the state of the EnsembleAnalysis object. Listing 2.3 showcases different feature extraction functionalities.

```

# Perform feature extraction for C-alpha distances
analysis.extract_features(featurization="ca_dist")

# Access the extracted features stored in the EnsembleAnalysis object
ca_dist_features = analysis.features

# Perform feature extraction for phi and psi angles without changing the
    state of the EnsembleAnalysis object
phi_psi_features = analysis.get_features(featurization="phi_psi")

# Get a summary dataframe of selected features with variability
summary_df = analysis.get_features_summary_dataframe(selected_features=["
    rg", "asphericity"], show_variability=True)
```

Listing 2.3: Feature extraction example using the EnsembleAnalysis class. This demonstrates how to extract features, access them, perform extraction without changing the state, and obtain a summary dataframe with selected features and variability.

When developing IDPET, one of the goals was to anticipate future methods for feature extraction. To achieve this, feature extraction functions utilize the ”argument unpacking” approach. This method allows for an arbitrary number of positional and keyword arguments to be accepted and passed directly to various feature extraction functions. By doing so, it eliminates the need to explicitly define every possible parameter in the method’s signature, enhancing the flexibility and extensibility of the code.

IDPET offers an additional feature extraction option that outputs ensemble averages and

standard deviations of global feature distributions in a summary Pandas dataframe. This functionality takes a list of strings indicating the desired features to be extracted as input, along with a boolean parameter specifying whether to include standard deviations with the averages. If no features are specified, the function returns a predefined list of global features.

2.7 Ensemble Comparison

In addition to comparing average values and distributions of various global and local features, there is a need to quantify the similarity between entire ensembles with a single metric. To achieve this, IDPET utilizes techniques which compute the divergence between distributions of extracted local features. Specifically, two techniques are employed: the Jensen-Shannon Divergence (JSD) and the Earth Mover's Distance (EMD).These methods are utilized to compare the distributions of *Cα-Cα* distances and α angles, which were described previously.

Previous studies have applied these techniques in related contexts: JSD has been used to assess similarities among ensembles of CGRP (calcitonin gene-related peptide) variants based on *Cα-Cα* distance distributions [91], while the EMD has been employed to compare predicted structural ensembles with ground truth structures by relying on RMSD, termed EMD-RMSD [92].

2.7.1 Jensen–Shannon Divergence (JSD)

In probability theory and statistics, the Jensen–Shannon divergence (JSD) is a technique for assessing the similarity between two probability distributions. Although it is based on the Kullback–Leibler divergence, it is distinct in being symmetric and always producing a finite value.

The Kullback–Leibler Divergence (KLD) $KL : P \times P \to [0, \infty]$ is a fundamental measure of distance between probability distributions, defined by:

$$KL(P : Q) := \int p \log\left(\frac{p}{q}\right) d\mu,$$

where p and q represent the Radon–Nikodym derivatives of probability measures P and Q with respect to μ, a positive measure.

Generally, the Kullback–Leibler Divergence (KLD) is an asymetric measure of distance. One well-known approach to symmetrize the KLD is through the JSD:

$$JS(P \parallel Q) := \frac{1}{2}\left(KL\left(P : \frac{P+Q}{2}\right) + KL\left(Q : \frac{P+Q}{2}\right)\right),$$

which can also be written as:

$$JS(P \parallel Q) := \frac{1}{2}\int \left(p \log\left(\frac{2p}{p+q}\right) + q \log\left(\frac{2q}{p+q}\right)\right) d\mu.$$

This distance can be interpreted as the overall divergence to the average distribution. An important characteristic of the JSD is that it is always bounded:

$$0 \leq JS(P \parallel Q) \leq \log 2.$$

Finally, the square root of the JSD ($\sqrt{JS}$) produces a metric distance that satisfies the triangular inequality [93].

2.7.2 Earth Mover's Distance (EMD)

The Earth Mover's Distance (EMD) is a technique used to measure the dissimilarity between two multi-dimensional distributions within a feature space, where a distance metric, known as the ground distance, is specified for individual features.

Intuitively, given two distributions, one can be seen as a mass of earth spread in space, and the other as a collection of holes in the same space. The EMD measures the least amount of work required to fill the holes with the earth. In this context, a unit of work is defined as moving a unit of earth over a unit of (ground) distance.

Computing the EMD is based on solving the transportation problem. Let I represent a set of suppliers, J a set of consumers, and c_{ij} the cost of transporting a unit of supply from $i \in I$ to $j \in J$. The objective is to determine a set of flows f_{ij} that minimize the total transportation cost:

$$\min \sum_{i \in I} \sum_{j \in J} c_{ij} f_{ij}$$

Each distribution can be represented by a signature, which consists of a set of clusters. Each cluster is characterized by its mean (or mode) and the fraction of the distribution that it contains.

The transportation problem can be effectively applied to match signatures by treating one

signature as the supplier and the other as the consumer. In this context, the cost c_{ij} represents the ground distance between element i in the first signature and element j in the second signature. After solving the transportation problem and finding the optimal flow f_{ij}, the EMD is determined by dividing the total transportation cost by the total flow:

$$EMD(x, y) = \frac{\sum_{i \in I} \sum_{j \in J} c_{ij} f_{ij}}{\sum_{i \in I} \sum_{j \in J} f_{ij}}$$

The normalization factor is introduced to prevent bias towards signatures with smaller total weights [94].

2.8 Dimensionality Reduction and Clustering

IDPET incorporates several dimensionality reduction techniques into its data analysis pipeline to handle the high-dimensional nature of IDP ensembles. These techniques enable the projection of data into a two-dimensional space for visualization purposes. By doing so, they uncover patterns in the conformations of IDPs and provide detailed insights into conformational substates when combined with clustering methods.

Dimensionality reduction pipelines are not applied directly to the Cartesian coordinates of each atom. Instead, they operate on features extracted from these trajectories, such as $C\alpha$-$C\alpha$ distances or φ and ψ angles, which contain more significant information about protein structures. Implemented dimensionality reduction methods include T-distributed Stochastic Neighbor Embedding (t-SNE), Principal Component Analysis (PCA), Kernel Principal Component Analysis (KernelPCA), the Force Scheme with Dimenfix modification, and Uniform Manifold Approximation and Projection (UMAP).

2.8.1 T-distributed Stochastic Neighbor Embedding (t-SNE)

t-Distributed Stochastic Neighbor Embedding (t-SNE) is a statistical method designed to visualize high-dimensional data by mapping each data point onto a lower dimensional space. It was created as an improvement over the original Stochastic Neighbor Embedding (SNE) algorithm.

At a high level, the SNE algorithm starts by converting the high-dimensional Euclidean distances between datapoints into conditional probabilities that represent similarities. Next, SNE defines an analogous conditional probability for the low-dimensional features and attempts to obtain the low-dimensional data representation that minimizes the Kullback-Leibler (KL) di-

vergence between the two probabilities. To achieve this, SNE minimizes the sum of Kullback-Leibler divergences over all datapoints using a gradient descent method.

The cost function employed by t-SNE differs from SNE in two main aspects: it employs a symmetrized version of the SNE cost function with simpler gradients, and it uses a Student-t distribution instead of a Gaussian to determine the similarity between two points in the lower-dimensional space [95, 96].

IDPET addapts the approach in [47], which combines t-SNE dimensionality reduction with k-means clustering to visualize complex IDP data and highlight conformational substates. The fundamental approach involves clustering the heterogeneous mixture of disordered protein conformations into subsets characterized by unique homogeneous structures.

This approach requires tuning two hyperparameters to achieve optimal results. The first hyperparameter represents the number of clusters assigned to the transformed data using k-means. The second hyperparameter is perplexity, associated with t-SNE, which determines how to balance attention between local and global aspects of the data. It essentially estimates the number of close neighbors for each point. Perplexity has a complex effect on t-SNE performance, with typical values ranging from 5 to 50.

To determine the best hyperparameters, IDPET evaluates different combinations of cluster numbers and perplexity values. The method selects the combination that results in the most interpretable clusters, as determined by the Silhouette score. This score measures the quality of clustering by evaluating both the cohesiveness of each point within its cluster (its proximity to the cluster center) and its separateness from points in other clusters (its distance from other cluster centers). The Silhouette score ranges from -1 to 1, with positive values indicating good clustering quality.

Since the clusters are determined on the reduced t-SNE data, computing the Silhouette score on this low-dimensional representation alone can be misleading. Therefore, the Silhouette score is also measured on the original high-dimensional features (S_{hd}) in addition to the score computed on the low-dimensional data produced by the algorithm (S_{ld}). The integrated score, calculated as the product of these two scores ($S_{ld} * S_{hd}$), has proven to be a reliable metric for assessing cluster quality.

IDPET relies on the scikit-learn implementation of t-SNE, which uses a specified metric to calculate distances between instances in a feature array. While IDPET can use the standard metrics defined by scikit-learn, it also offers an additional metric for circular data such as φ and ψ torsion angles, where Euclidean distances are not effective. The "circular" option employs a custom metric, unit vector distance. This metric converts the angles into unit vectors in 2D

space by representing each angle with its sine and cosine values and then computes the sum of distances between these unit vectors.

2.8.2 Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is possibly the oldest and most popular multivariate statistical technique. The goal of PCA is to extract important information from the data by expressing it as a set of new orthogonal variables called principal components. These principal components are linear combinations of the original features in the dataset. The first principal component is computed to capture the largest possible variance in the data. The second principal component is computed to be orthogonal to the first and to capture the next largest variance. Subsequent components are computed similarly. The values of the principal components for the observations are called factor scores, which can be interpreted geometrically as the projections of the observations onto the principal components [97].

PCA maps a data vector from its original set of variables to a new set of uncorrelated variables. However, not all principal components need to be retained. Dimensionality reduction is achieved by keeping only the first few principal components that capture the most significant variance in the data.

As a linear dimensionality reduction method, PCA has limited performance in retaining local neighborhoods. However, IDPET includes PCA as a baseline and reference point, providing functionalities for PCA dimensionality reduction alongside more advanced non-linear methods.

2.8.3 Kernel Principal Component Analisis (KernelPCA)

Kernel Principal Component Analysis (Kernel PCA) is a method for performing a nonlinear form of Principal Component Analysis (PCA). This modification is achieved by generalizing PCA to handle features that are nonlinearly related to the input variables. While traditional PCA projects data onto a lower-dimensional subspace, Kernel PCA implicitly maps the data to a higher-dimensional feature space using a kernel function. This allows for the computation of principal components in this higher-dimensional space without the need to explicitly compute the coordinates in that space [98].

In the context of IDPET, Kernel PCA addresses the limitations of linear dimensionality reduction methods, particularly when dealing with circular data like φ and ψ torsion angles. For such data, extreme values like -180° and 180° are practically adjacent, posing a challenge

for linear methods. Kernel PCA, as a nonlinear variant of PCA, more effectively preserves the intrinsic relationships between these circular data points, ensuring that their true proximity is maintained in the reduced dimensional space.

2.8.4 Force Scheme with Dimenfix

The nearest-neighbor projection (NNP) is a dimensionality reduction method that projects multi-dimensional data into a two-dimensional space intended for visualization. The fundamental approach to NNP is straightforward: for each new data point to be projected, two nearest neighbors among projected data points are used to establish its position in the projected space. The position of the new point $x' = \alpha(x)$ in the projected space is determined by the intersection of two circles centered at $q' = \alpha(q)$ and $r' = \alpha(r)$, with radii equal to $d(x, q)$ and $d(x, r)$, respectively, where d refers to the distance criterion in the original space. If there is no intersection between the circles, an intermediary point between their centers is used (Figure 2.6). The outcome of this approach is a projection that preserves local neighbors and offers information about relationships between instances in the original space.

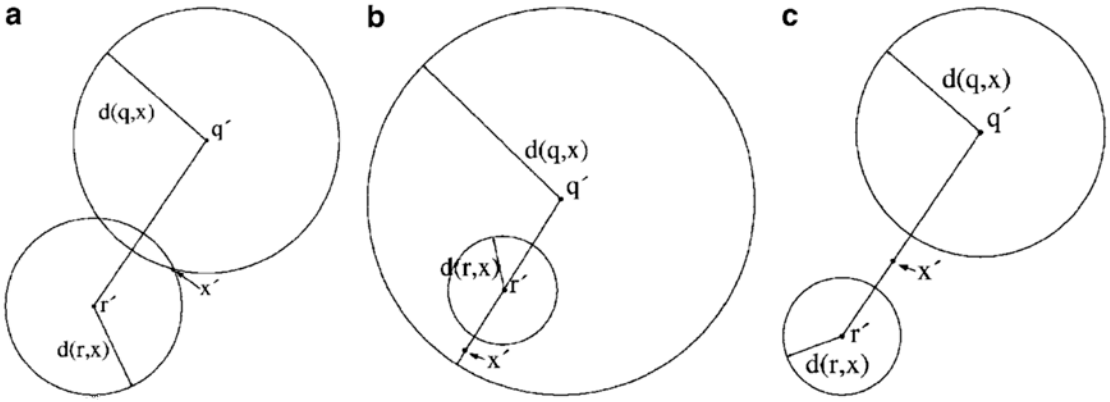

Figure 2.6: Intersection cases for NNP mapping. (a) Circles intersect. (b) No intersections, one internal circle. (c) No intersection or inclusion. This figure was taken from [13].

These mappings can be further improved by applying normalization to the coordinates of the projected instances in the two dimensional space. This improvement scheme is called the Force approach, referring to the concept that points are ”attracted to” or ”repelled by” each other. In the algorithm, Δ approximates the difference between the projected distance and the distance in the original space. This approximation is defined as:

$$\Delta = \frac{d(x,q) - d_{\min}}{d_{\max} - d_{\min}} - d_2(x', q')$$

Here, $d(x,q)$ represents the distance criterion in the original n-dimensional space between points x and q, while $d_{\min}$ and $d_{\max}$ denote the minimum and maximum distances in that space, respectively. x' and q' are the projected data points obtained after transformation, denoted as $x' = \alpha(x)$ and $q' = \alpha(q)$. $d_2(x', q')$ refers to the distance criterion in the transformed space.

The Force Scheme enhances proximity relationships in the projected space by reducing the distance between points that were placed farther apart than their original distances. Importantly, the Force Scheme can be applied to adjust projections produced by any dimensionality reduction method [13].

DimenFix is a meta-method that can be applied to any dimensionality reduction technique involving gradient descent-like steps. It preserves the values of a specific feature during dimensionality reduction and controls changes in relationships between points concerning the selected feature to a specified degree. DimenFix operates in two modes: the Strictly Fixed mode, which prevents a point from moving along the fixed axis, and the Moving-in-Range mode, which allows a point to move within a defined range [99].

IDPET integrates the DimenFix-modified Force Scheme with k-means clustering to identify representative structural features in the visualization of conformational ensembles.

2.8.5 Uniform Manifold Approximation and Projection (UMAP)

Non-linear dimensionality reduction algorithms like t-SNE are highly effective at preserving features in high-dimensional data. The Uniform Manifold Approximation and Projection (UMAP) algorithm is competitive with t-SNE, which is the current state-of-the-art for visualization. UMAP arguably preserves more of the global structure and offers superior performance.

At a high level, UMAP approximates local manifolds and combines their local fuzzy simplicial set representations to build a topological representation of high-dimensional data. This process can similarly be applied to a low-dimensional representation of the data to construct an equivalent topological representation. UMAP then optimizes the layout of the data in the low-dimensional space by minimizing the cross-entropy between the two topological representations.

From a computational standpoint, UMAP can be classified as a k-neighbor based graph learning algoritm along with t-SNE. Like other k-neighbor graph-based algorithms, UMAP

operates in two phases. In the first phase, it constructs a specific weighted k-neighbor graph. In the second phase, it computes a low-dimensional layout of this graph. The differences among algorithms in this class lie in the specific methods used to construct the graph and compute the layout.

The UMAP algorithm, requires four hyperparameters: the number of neighbors to consider when approximating the local metric, the target embedding dimension, the desired separation between close points in the embedding space, and the number of training epochs used to optimize the low-dimensional representation [100].

IDPET incorporates UMAP into its dimensionality reduction analysis routines, following a simplified approach described in [101]. By applying UMAP to extracted features such as backbone torsion angles or *Cα-Cα* distances and combining it with k-means clustering, it allows for a comparative analysis with other dimensionality reduction methods like t-SNE through a uniform interface. As with t-SNE, IDPET allows users to specify multiple values for the number of neighbors and the number of clusters when executing UMAP. For each combination of hyperparameters, the silhouette score is computed for the resulting clusters, and the combination with the highest score is retained.

2.8.6 Dimensionality Reduction Workflow

The dimensionality reduction workflow in IDPET follows a two-step procedure. First, local features are extracted, such as *Cα-Cα* distances, φ and ψ torsion angles, α angles, and trRosetta φ and ω angles. Next, dimensionality reduction is performed on the EnsembleAnalysis object by calling a single function (`reduce_features`) and providing the chosen method along with method-specific parameters. The function returns the transformed data as a NumPy array, and also stores it within the EnsembleAnalysis class for easy access and visualization at any time. The two-step approach was chosen to allow different dimensionality reduction algorithms to be tested on the same extracted features, enabling the best results to be obtained without needing to recompute the features for each execution (Listing 2.4).

2.9 Visualization

The visualization functions in IDPET are provided by the `visualization` module, which includes the `Visualization` class. This class is instantiated with an `EnsembleAnalysis` object that encompasses all data generated during various analysis steps. This data includes trajectories of the loaded

```

# Perform feature extraction for torsion angles
analysis.extract_features(featurization="phi_psi")

# Perform dimensionality reduction using t-SNE
analysis.reduce_features(method='tsne', perplexity_vals=[10, 20, 50],
    circular=False, range_n_clusters=range(2, 10, 1))
```

Listing 2.4: Dimensionality Reduction Workflow Example

ensembles, extracted features, reduced data, and metadata, which are essential for the diverse tools available within the Visualization class.

Most plotting functions in the Visualization class are closely related to one of the analysis routines: local, global, ensemble comparison, or dimensionality reduction. These functions are designed to be both customizable and extendable. They accept a Matplotlib Axes object or a list of Axes objects if multiple plots are displayed, allowing for the creation of publication-quality plots with custom layouts.

Many functions offer unique parameters for further customization or provide multiple representations of the same feature. Additionally, all supported plot functions include a boolean save parameter, which determines whether the resulting plots should be saved in the output directory of the Visualization class in PNG format. To facilitate extendability, all functions return their Axes objects after plotting. An example of a visualization function for analyzing asphericity is shown in Listing 2.5.

```
# Instantiate the Visualization class with the analysis object
visualization = Visualization(analysis=analysis)

# Generate an asphericity plot with specific parameters:
# - bins: Number of bins for the histogram (10)
# - violin_plot: Use a violin plot representation (True)
# - location: Statistical measure to display ('median')
# - save: Save the plot to the output directory (True)
visualization.asphericity(bins=10, violin_plot=True, location='median',
    save=True)
```

Listing 2.5: Generating and Saving an Asphericity Plot

2.10 Documentation and Installation

IDPET features extensive documentation generated using Sphinx. Sphinx's sphinx-quickstart tool simplifies initial setup by creating a documentation source directory with a configuration file (conf.py) and a master document (index.rst). The master document serves as a welcome page and contains the root of the "table of contents tree" (toctree). Additional source files in re-StructuredText (rst) format showcase IDPET's capabilities, linked hierarchically from the master document. Sphinx's autodoc extension includes docstrings from the package modules, providing a comprehensive description of IDPET's core functionalities. The final documentation, consisting of HTML pages, is generated using a Makefile and make.bat file.

The documentation (Figure 2.7) consists of an overview, which outlines the primary use case of the package, an Installation page, and a comprehensive demo showcasing global analysis, local analysis, dimensionality reduction, and ensemble comparison. Additionally, it provides detailed documentation on various functions distributed across the ensemble_analysis, ensemble, and visualization modules.

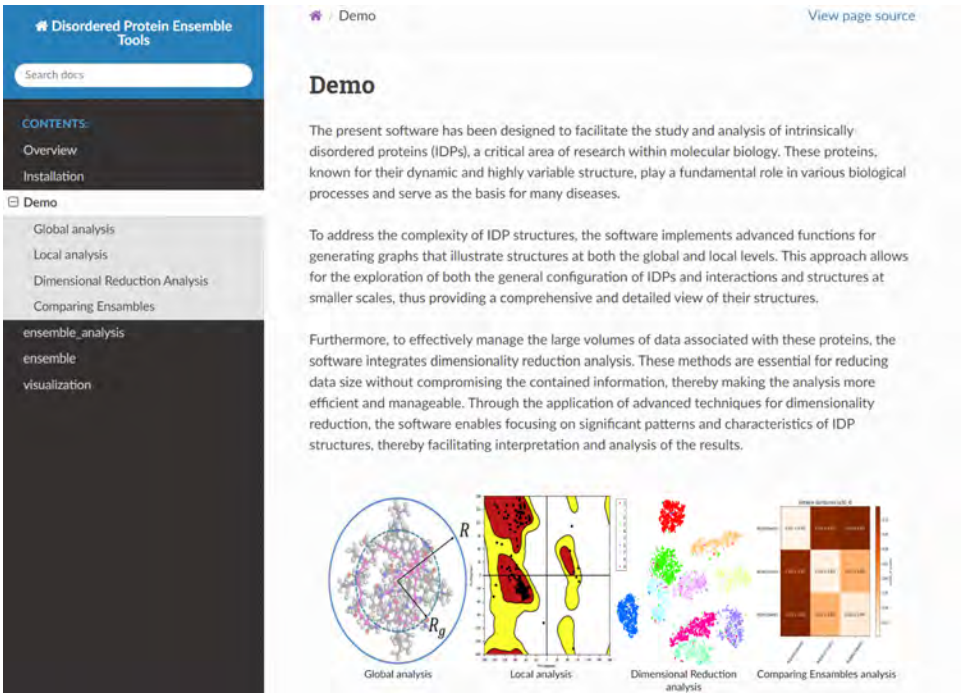

Figure 2.7: IDPET's documentation includes an overview of the package's basic use, installation instructions, a detailed demo covering various analyses, and descriptions of functions within the ensemble_analysis, ensemble, and visualization modules.

IDPET is available for download through the Python Package Index (PyPI) and Conda. PyPI is a repository for Python software packages. The process of uploading a project to PyPI involves several steps. First, the project directory must be structured appropriately, with Python files organized in a directory named to match the project name. The pyproject.toml file specifies the build backend, such as setuptools, which is responsible for converting the source code into distribution packages. Additional configurations, including metadata and dependencies, are included in a setup.cfg file. The build process generates two types of distribution files: a source distribution (.tar.gz) and a built distribution (.whl). These packages are then uploaded to PyPI using Twine and can be installed with pip.

Conda is a powerful command-line tool for package and environment management. To build a package with Conda, three files are required: a meta.yaml file containing metadata such as the required Python version and dependencies; build.sh, a shell script for macOS and Linux; and bld.bat, a batch file for Windows. For IDPET, the build process uses PyPI as the source by providing the package URL and SHA-256 hash. After building the package with conda-build for the current platform, it is converted to be compatible with other platforms. The resulting packages are then uploaded to Anaconda.org, making them available for installation using conda.

3 Results

3.1 Use Case: drkN SH3 Domain Analysis

In this section, the goal is to demonstrate how the IDPET framework can enhance the analysis of conformational ensembles of IDPs, using a practical example.

Lincoff et al.[102] examined the impact of the initial conformation pool on the derived ensemble, focusing on the unfolded state of the N-terminal *SH*3 domain of Drosophila Drk, consistent with previously published NMR, SAXS, and smFRET data. Three different conformation pools were used in the study: a completely random pool generated using TraDES [28], an experimentally restrained pool obtained with ENSEMBLE [20], and a third pool consisting of an equal mixture of the first two pools. Using the Bayesian-based selection method X-EISD [103], three distinct structural ensembles were derived from these starting pools. The analysis showed that the ensemble from the mixed pool best matched experimental restraints across multiple data sets. Throughout the analysis, the distribution of the radius of gyration was used to gain insights into the protein's overall size and shape, and secondary structure propensity data were examined to identify changes in the features of the three ensembles.

To showcase the effectiveness of IDPET, the ensembles obtained in the study from the different pools of conformers are compared against one another. These ensembles are available in the Protein Ensemble Database (PED) under the following IDs: PED00156 for the ensemble generated from the random pool, PED00157 for the ensemble obtained from the experimen-

tally constrained pool, and PED00158 for the ensemble derived from the mixed pool. Despite representing the same protein, these structural ensembles exhibit significant variation in their conformations. With IDPET, there's no need for manual download of these ensembles. The package utilizes the PED API to automatically download and load the ensemble files by specifying their IDs in the appropriate format.

The analysis of results is divided into four main sections: Global Analysis, Local Analysis, Ensemble Comparison, and Dimensionality Reduction.

3.2 Global Analysis

The original paper exclusively used the radius of gyration and secondary structure propensities to compare the Random, Mixed, and Experimental ensembles. IDPET not only simplifies the extraction and visualization of these features but also analyzes other global features of the ensembles, providing comprehensive insights into their comparison.

First, IDPET can be used to obtain a summary of the ensemble's global features in a Pandas DataFrame format (Table 3.1). The table shows that while the mean values of many global features are quite similar, the standard deviations indicate differences in the underlying distributions of the conformation features. Therefore, visualizing these distributions is essential to gain a more comprehensive understanding of the structures.

ensemble_code	n_residues	n_conformers	rg_mean	rg_std	end_to_end_mean	end_to_end_std	asphericity_mean	asphericity_std	sasa_mean	sasa_std
PED00156e001	59	100	1.79	0.59	4.33	2.39	0.37	0.27	65.25	8.24
PED00157e001	59	100	1.91	0.20	4.34	1.33	0.48	0.14	64.02	4.08
PED00158e001	59	88	1.87	0.19	4.38	1.17	0.53	0.13	62.22	4.57

Table 3.1: Summary of global features for ensembles PED00156e001, PED00157e001, and PED00158e001. The table presents mean and standard deviation values for radius of gyration , end-to-end distance, asphericity, and solvent accessible surface area (SASA).

The figure (Figure 3.1) was created using IDPET with a layout consisting of a two-by-two subplot grid. Each subplot visualizes specific features of the three ensembles (PED00156, PED00157, and PED00158). The subplot located at the top-left displays the distribution of radius of gyration using violin plots colored in blue. The top-right subplot illustrates the distribution of end-to-end distances, represented in yellow. The bottom-left subplot shows the distribution of asphericity values depicted in green. Lastly, the bottom-right subplot presents the distribution of global SASA values in orange. Horizontal lines in each subplot indicate the mean values of their respective distributions.

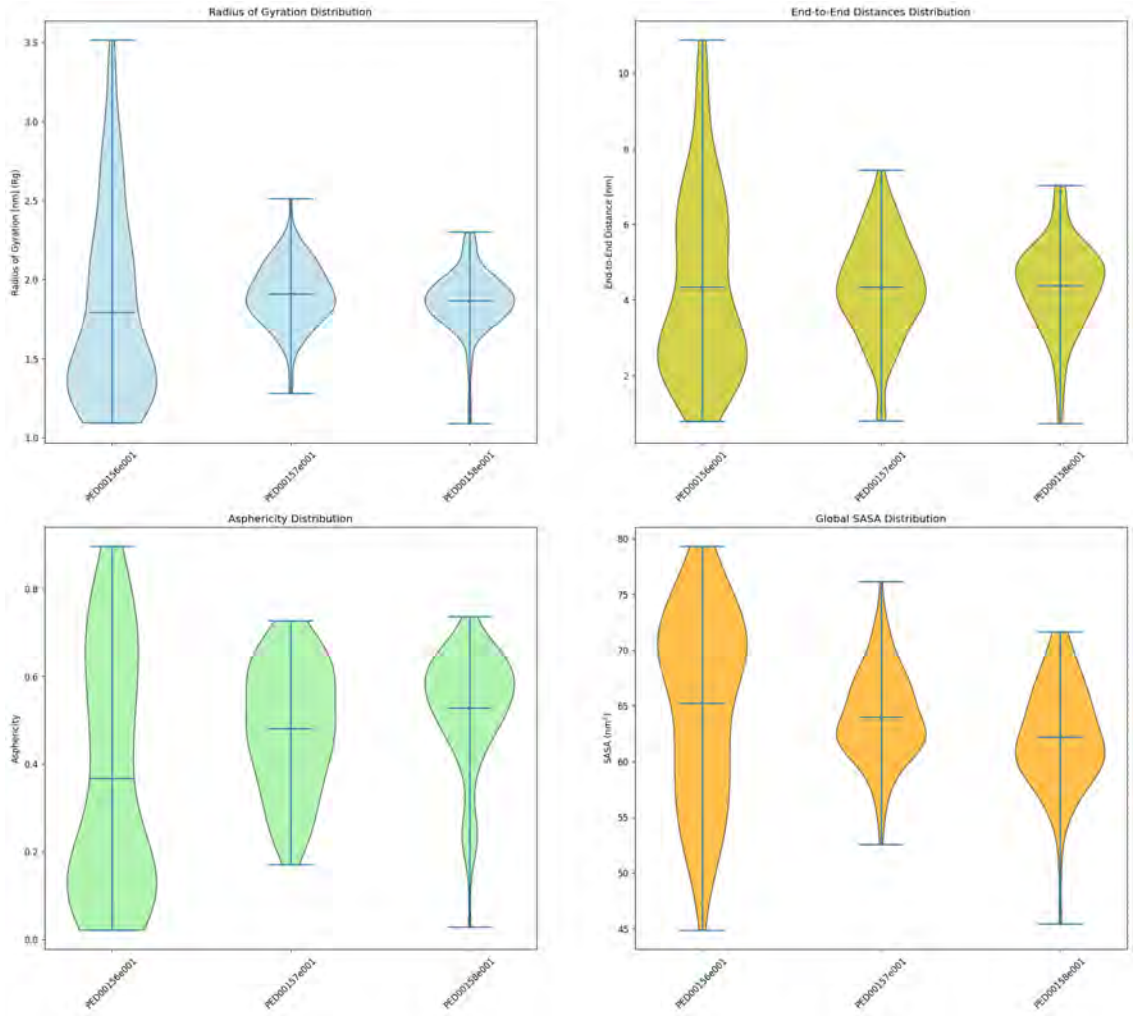

Figure 3.1: Distributions of radius of gyration, end-to-end distances, asphericity, and global SASA for ensembles PED00156, PED00157, and PED00158, visualized using IDPET. Each subplot corresponds to a specific feature, with radius of gyration shown in blue, end-to-end distances in yellow, asphericity in green, and SASA in orange. Horizontal lines indicate the mean values of each distribution.

Although the ensemble means for the radius of gyration across all conformations are similar, the distributions of these values differ significantly. The ensemble derived from the random pool (PED00156) exhibits a broader range of values and a less uniform distribution compared to the ensembles from the experimental (PED00157) and mixed (PED00158) pools. The presence of values in the higher range suggests the existence of highly extended conformations within the random ensemble. In contrast, the presence of secondary structures in the conformers of the latter two ensembles contributes to their relatively compact structures, as indicated

by their radius of gyration distributions. Since there is a high correlation between radius of gyration and end-to-end distances, the distributions are nearly identical.

A conformation-level measure of SASA is obtained by summing the SASA values computed for every residue within a conformation. The distribution of this global SASA is then plotted for each ensemble. The plots reveal a direct correlation between radius of gyration and global SASA. The narrower distributions observed in the experimental (PED00157) and mixed (PED00158) ensembles can be attributed to the higher presence of secondary structure elements. This suggests that the ENSEMBLE method, used to derive the experimentally constrained structural ensembles, is highly biased towards experimental data such as NMR.

Distributions of asphericity plotted for each of the three ensembles quantify the deviation of individual conformations from a perfect sphere, with lower values indicating a more spherical structure. As with previous features, the random ensemble (PED00156) exhibits a wider distribution, encompassing extreme shapes ranging from nearly perfect spheres to nearly perfect rods. In contrast, the shapes in the experimental (PED00157) and mixed (PED00158) ensembles are much more uniform. Notably, the mean asphericity is significantly lower for the random ensemble, indicating a greater presence of more spherical structures within this set.

The diversity in global properties of IDPs, such as size and shape, is often obscured by ensemble-average properties. Solely consulting the average properties can lead to the assumption that the analyzed ensembles are very similar, when they are, in fact, quite different. Analyzing the simulation trajectories of the ensembles can reveal these differences through the distributions of conformation properties.

The relationship between radius of gyration and asphericity provides another insightful metric worth examining (Figure 3.2). The random ensemble (PED00156) exhibits a strong positive correlation between radius of gyration and asphericity, indicating that the larger structures in the ensemble also tend to be less spherical. Conversely, the experimental ensemble (PED00157) is characterized by a weaker correlation, which can be attributed to a greater presence of secondary structures. The mixed ensemble shows an intermediate correlation, as expected.

The Flory exponent describes how the radius of gyration of a polymer chain scales in a specific solvent environment. Using IDPET, the Flory exponent is computed for the analyzed ensembles. The computed Flory exponents for the ensembles are summarized in Table 3.2. These results indicate that PED00156 has the most expanded conformations among all three ensembles, irrespective of radius of gyration, according to polymer physics principles. The

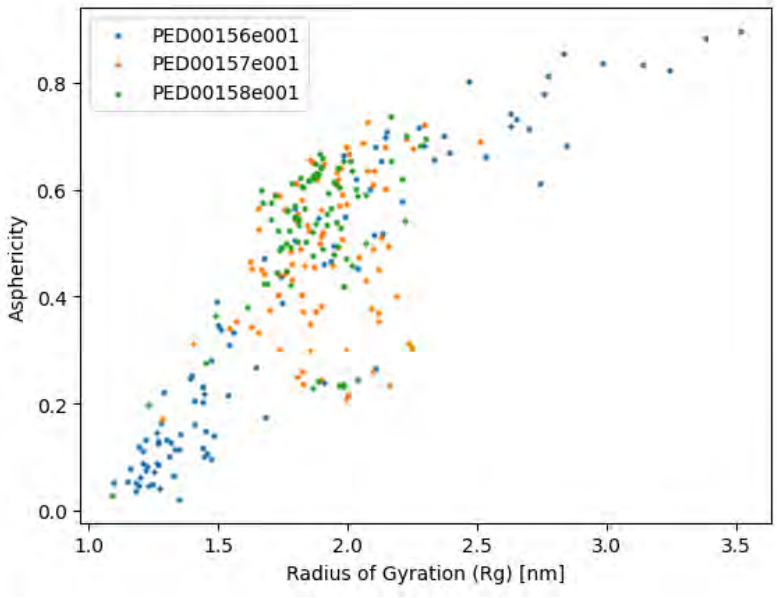

Figure 3.2: The relationship between radius of gyration and asphericity across ensembles (PED00156, PED00157, and PED00158) demonstrates varying correlations. The random ensemble (PED00156) shows a strong positive correlation, indicating larger structures are less spherical. The experimental ensemble (PED00157) exhibits a weaker correlation, attributed to higher secondary structure content, while the mixed ensemble shows an intermediate correlation.

lower values of the exponent in the other two ensembles can be attributed to the presence of secondary structures.

Ensemble	Flory Exponent
PED00156	0.600
PED00157	0.555
PED00158	0.573

Table 3.2: Flory exponents computed for the analyzed ensembles using IDPET.

3.3 Local Analysis

In this section, the analysis begins with the replication of the secondary structure propensity analysis from the original paper to demonstrate the effectiveness of IDPET. Following this, an analysis of other local features is conducted, including site-specific flexibility, 2D Ramachandran histograms, and alpha angle distributions.

The figure (Figure 3.3) was created using IDPET with a layout consisting of three vertically stacked subplots. Each subplot visualizes specific features of the three ensembles (PED00156, PED00157, and PED00158). The top subplot displays the relative DSSP content for helices ('H'), indicating the propensity of helical structures across residue positions in the analyzed ensembles. The middle subplot illustrates the site-specific flexibility parameter, highlighting regions of increased flexibility. The bottom subplot presents the distribution of alpha angles, describing the geometric relationships between adjacent segments of the protein backbone.

The IDPET function for displaying secondary structure propensities utilizes MDTraj's implementation of the DSSP algorithm, supporting letter codes such as 'H' for helices, 'C' for coils, and 'E' for strands. The plot showing the relative presence of helices per residue reveals significant differences among the ensembles PED00156, PED00157, and PED00158. PED00157 and PED00158 are characterized by several regions with a distinctly high presence of helices, specifically at residue positions 16-20, 30-45, and 50-55. In contrast, the random ensemble (PED00156) lacks any meaningful presence of helices. Additionally, the similarity in the secondary structure patterns of PED00157 and PED00158 suggests that these ensembles contain comparable structures.

Site-specific flexibility provides insight into local disorder by analyzing the circular variance of the Ramachandran φ and ψ angles for each residue position. The values of this parameter range from 0, indicating identical angles across all conformations, to 1, representing a uniform distribution. According to this metric, PED00156 exhibits higher flexibility in two regions, specifically residues 16-20 and 30-45.

Alpha angles are defined between two hyperplanes formed by four consecutive $C\alpha$ atoms and are used to describe the geometric relationship between adjacent segments of the protein backbone. Visualizing the density distribution of alpha angles helps quantify these relationships at the ensemble level. Angles ranging between 50° and 70° (approximately 1 radian) correspond to helical backbone geometries. The higher densities exhibited by PED00157 and PED00158 around this range indicate the presence of these structures in the ensembles. These results align with previous findings.

The 2D Ramachandran histogram displays φ and ψ angle distributions, indicating secondary structures in the protein. The Ramachandran diagrams for the three analyzed ensembles (Figure 3.4) show a higher density in the helical region (between -90° and -40°) for PED00157 and PED00158 compared to PED00156. This observation is consistent with the higher helicity previously determined in these ensembles.

Figure 3.3: Visualization of local features for three ensembles (PED00156, PED00157, and PED00158) created using IDPET. The top subplot shows the relative DSSP content for helices ('H'), indicating the helical structure propensity across residues. The middle subplot illustrates the site-specific flexibility parameter, highlighting regions of increased flexibility around residues 16-20 and 30-45. The bottom subplot presents the distribution of alpha angles, describing the geometric relationships between adjacent segments of the protein backbone.

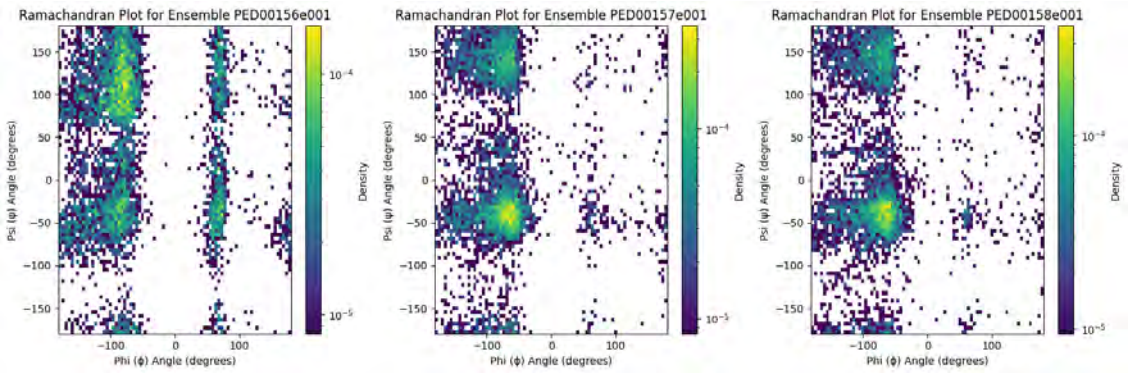

Figure 3.4: 2D Ramachandran histograms of φ and ψ angle distributions for three ensembles (PED00156, PED00157, and PED00158). Higher density in the helical region (between -90° and -40°) for PED00157 and PED00158 indicates greater helicity compared to PED00156.

3.4 Ensemble Comparison

Using IDPET, pairwise comparisons between ensembles can be performed based on the distributions of extracted features. Two methods for measuring the divergence between these distributions are supported: the Jensen-Shannon Divergence (JSD) and the Earth Mover's Distance (EMD). The outcome of these methods is a numerical score ranging from 0, indicating identical distributions, to 1, indicating complete divergence.

In this analysis, the Jensen-Shannon Divergence is employed to compare the distributions of *Cα-Cα* distances and α angles between the analyzed ensembles. A plot with two subplots was created, each displaying a colored comparison matrix. One subplot compares the distribution of *Cα-Cα* distances, while the other compares the distribution of α angles using Jensen-Shannon Divergence.

The results (Figure 3.5) clearly illustrate the differences between the three ensembles. As anticipated, PED00156 shows greater divergence from PED00157 and PED00158 compared to the divergence observed between the latter two, both in terms of distributions of *Cα-Cα* distances and α angles.

3.5 Dimensionality Reduction Analysis

Dimensionality reduction techniques enable a detailed examination of structural features extracted from ensemble data. In IDPET, this process involves two main steps: extracting the specified features and then applying a dimensionality reduction algorithm, often in combina-

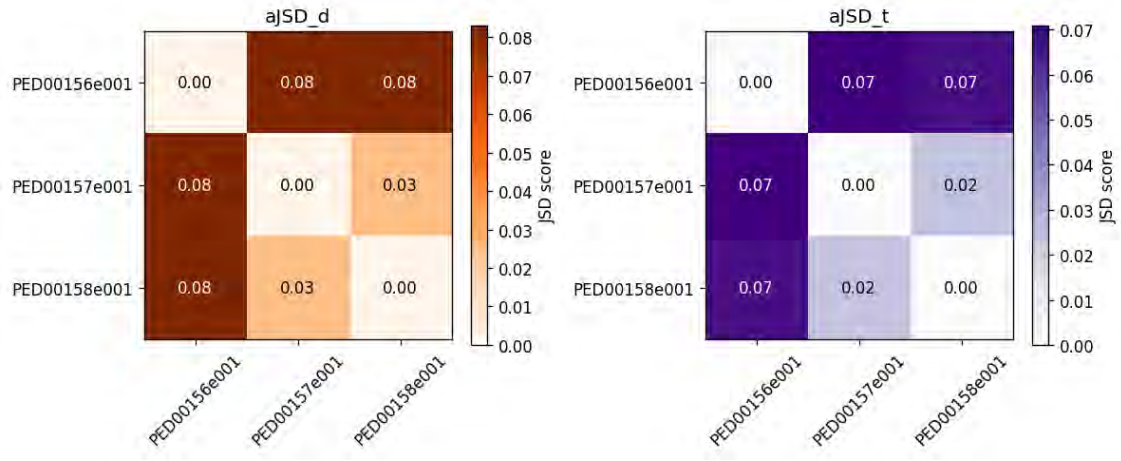

Figure 3.5: Comparison matrices of Jensen-Shannon Divergence for $C\alpha$-$C\alpha$ distances and α angles across three structural ensembles. The left matrix illustrates the divergence in $C\alpha$-$C\alpha$ distances, while the right matrix shows the divergence in α angles. The analysis highlights that PED00156 diverges more significantly from the other two ensembles compared to the divergence between the latter two.

tion with clustering methods. This section demonstrates the utility of this approach using the three ensembles from the original paper. Two sets of features, φ and ψ angles and $C\alpha$-$C\alpha$ distances, will be analyzed by applying the t-distributed stochastic neighbor embedding (t-SNE) algorithm along with k-means clustering.

t-SNE is the current state-of-the-art technique for visualizing high-dimensional data. When combined with k-means clustering, two hyperparameters need to be specified: perplexity and the number of clusters. Perplexity balances attention between global and local aspects of the data in t-SNE, while the number of clusters is used in k-means clustering. For both types of analyzed features, the algorithm was executed with perplexity values of 10, 20, 50, and 100, and a range of cluster numbers from 2 to 10. For each combination of parameters, the silhouette score, which measures the quality of the final clusters, was computed for both the original high-dimensional data and the transformed low-dimensional data. The best results were retained based on the product of these two scores. When executing t-SNE on extracted φ and ψ features, it is necessary to specify that the algorithm is working with angles. This modifies the distance function to first compute unit vectors from the angles before calculating the distances.

The results are visualized using a supported function that presents the conformations in the reduced space with different labels. The first plot retains the original labels, linking the conformations to their respective ensembles. The second plot displays the clustering labels, and the third plot can be specified to represent conformations using global features such as

radius of gyration, asphericity, SASA, etc. The fourth is a density plot with original labels.

After applying t-SNE and k-means clustering to extracted *Cα-Cα* distances, the best results were achieved with a perplexity of 100 and 3 clusters. The computed silhouette scores were 0.533 for the low-dimensional data and 0.253 for the original high-dimensional data (Table 3.3). Both positive scores indicate that the clusters are acceptable. The best score was selected based on the product of these two scores.

Feature Set	Perplexity	Number of Clusters	Silhouette Score (Low-Dim)	Silhouette Score (High-Dim)
Cα-Cα Distances	100	3	0.533	0.253
φ and ψ Angles	100	2	0.41	0.03

Table 3.3: Summary of t-SNE and k-means clustering results

Based on the number of clusters observed in the scatter plot with cluster labels, three conformational substates can be identified within the protein based on *Cα-Cα* distances (Figure 3.6). Analyzing the transformed features, the plot with original labels and the density plot reveal that conformations from PED00157 and PED00158 are similar based on carbon alpha inter-atomic distances. The radius of gyration labels plot illustrates how dimensionality reduction based on *Cα-Cα* distances arranges conformations from the lowest radius of gyration (bottom left) to the highest radius of gyration (top right). Additionally, it shows that some PED00156 conformations are similar to those in PED00157 and PED00158 in terms of inter-atomic *Cα-Cα* distances. This similarity is expected, given that the final mixed (PED00158) ensemble contains 24% of conformations from the random pool.

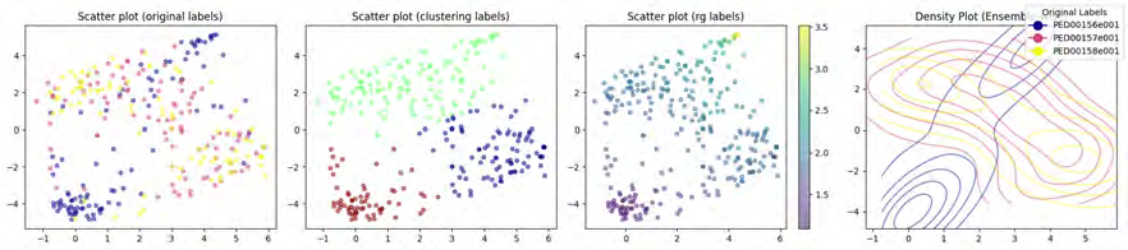

Figure 3.6: t-SNE and k-means clustering results based on *Cα-Cα* distances, with a perplexity of 100 and 3 clusters. The plots reveal similarities between PED00157 and PED00158 conformations and show how dimensionality reduction organizes conformations by radius of gyration. Some PED00156 conformations are similar to those in PED00157 and PED00158, reflecting the mixed nature of PED00158.

When t-SNE and k-means clustering were applied to φ and ψ angles, the best silhouette scores of 0.41 for low-dimensional data and 0.03 for high-dimensional data were obtained with a perplexity of 100 and 2 k-means clusters. While the resulting clusters were satisfactory, they

indicate the presence of two conformational substates within the protein, which differs from the results obtained using *Cα-Cα* distances. Additionally, the plots using original labels reveal that PED00156 is quite different from PED00157 and PED00158 in terms of dihedral angle distributions (Figure 3.7).

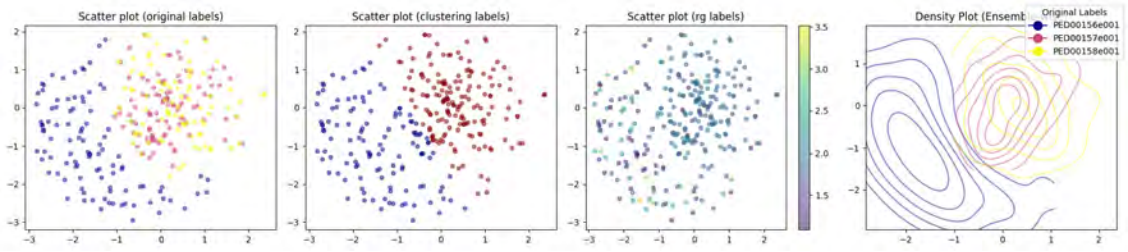

Figure 3.7: t-SNE and k-means clustering results based on φ and ψ angles, with a perplexity of 100 and 2 clusters. The plots indicate the presence of two conformational substates within the protein and show that PED00156 differs significantly from PED00157 and PED00158 in terms of dihedral angle distributions.

4

Conclusion

Summary of Work

This thesis describes the development and application of IDPET (Intrinsically Disordered Protein Ensemble Toolkit), an integrative Python software package designed for analyzing molecular simulations of intrinsically disordered proteins (IDPs). IDPET aims to be user-friendly and easy to install, achieved through an intuitive package structure and a consistent object-oriented interface. It features comprehensive documentation generated with Sphinx and supports quick installation via PyPI and Conda. IDPET is versatile, suitable for analyzing both all-atom and coarse-grained ensemble simulations, though the latter may require some modifications and has certain limitations.

IDPET was developed as an extensible framework that builds upon the general-purpose simulation analysis package MDTraj. While MDTraj focuses on well-defined proteins, IDPET enhances MDTraj's capabilities by specializing in the analysis of IDPs. IDPET is designed to handle multiple IDP ensembles simultaneously, supporting various file formats and remote interfaces from databases like PED and ATLAS.

IDPET offers a comprehensive range of analysis functionalities. It enables local analysis, focusing on residue-level features across all conformations within ensembles, and global analysis, which derives conformation-level averages to provide overall structural insights. The software also includes ensemble comparison methods that evaluate the divergence between distributions of features extracted from different ensembles. Additionally, IDPET employs dimensionality

reduction techniques to visualize ensemble features in two-dimensional space and cluster them, helping to identify distinct conformational substates. To achieve these analyses, IDPET integrates methodologies from various fields, including polymer physics.

Key Findings

Using IDPET, a comprehensive analysis of three structural ensembles of the unfolded drkN SH3 domain was conducted. These ensembles were derived from three conformation pools: a completely random pool generated using TraDES, an experimentally restrained pool obtained with ENSEMBLE, and a third pool consisting of an equal mixture of the first two pools. The analysis revealed that the latter two ensembles contained structures that were more compact and exhibited a higher presence of secondary structures. This was supported by both global and local feature analyses. Additionally, the latter two ensembles showed greater similarity concerning the analyzed features, as confirmed by ensemble comparison methods. Finally, dimensionality reduction combined with clustering suggested that up to three conformational substates can be defined for the observed protein.

Implications and Contributions

This study demonstrates that IDPET is an effective tool for analyzing structural ensembles of IDPs. By extending MDTraj's capabilities and offering specialized functions for IDP analysis, IDPET fills a significant gap in the current toolkit for protein analysis. A key contribution of IDPET is its ability to load multiple conformational ensembles from different data formats or remote databases directly, eliminating the need to preprocess or merge data into a single file. IDPET was developed as an independent package rather than an extension of MDTraj because its analysis routines are specifically tailored for disordered proteins and offer limited value for well-folded proteins. Additionally, IDPET aims to serve as a general platform for implementing novel analysis routines for disordered proteins, supporting the growing community of researchers in this field.

Limitations and Future Work

While IDPET is robust and versatile, its application to coarse-grained simulations requires further development to address current limitations. Although most analyses offered by IDPET can be applied to coarse-grained models, some may cause errors or yield meaningless results.

IDPET manages these cases appropriately but may not be the optimal tool for analyzing coarse-grained models. Developing an extension or a separate tool specifically designed for these models could provide a more effective solution.

IDPET was designed to be highly extendable, particularly in extracting various structural features and introducing dimensionality reduction and clustering methods. Future work could involve adding new methods to IDPET's analysis pipeline. Additionally, integrating IDPET into a web server could provide remote access to its functions, enhancing accessibility but potentially sacrificing some extensibility. Furthermore, given its close integration with remote database APIs, IDPET could be incorporated into PED, offering additional analysis options within the IDP database.

References

[1] C. I. Branden and J. Tooze, *Introduction to protein structure*. Garland Science, 2012.

[2] A. D. Baxevanis, G. D. Bader, and D. S. Wishart, *Bioinformatics*. John Wiley & Sons, 2020.

[3] R. Van Der Lee, M. Buljan, B. Lang, R. J. Weatheritt, G. W. Daughdrill, A. K. Dunker, M. Fuxreiter, J. Gough, J. Gsponer, D. T. Jones *et al.*, "Classification of intrinsically disordered regions and proteins," *Chemical reviews*, vol. 114, no. 13, pp. 6589–6631, 2014.

[4] J. Kragelj, V. Ozenne, M. Blackledge, and M. R. Jensen, "Conformational propensities of intrinsically disordered proteins from nmr chemical shifts," *ChemPhysChem*, vol. 14, no. 13, pp. 3034–3045, 2013.

[5] A. G. Kikhney and D. I. Svergun, "A practical guide to small angle x-ray scattering (saxs) of flexible and intrinsically disordered proteins," *FEBS letters*, vol. 589, no. 19, pp. 2570–2577, 2015.

[6] S. Bottaro and K. Lindorff-Larsen, "Biophysical experiments and biomolecular simulations: A perfect match?" *Science*, vol. 361, no. 6400, pp. 355–360, 2018.

[7] S. Kmiecik, D. Gront, M. Kolinski, L. Wieteska, A. E. Dawid, and A. Kolinski, "Coarse-grained protein models and their applications," *Chemical reviews*, vol. 116, no. 14, pp. 7898–7936, 2016.

[8] M. Bonomi, G. T. Heller, C. Camilloni, and M. Vendruscolo, "Principles of protein structural ensemble determination," *Current opinion in structural biology*, vol. 42, pp. 106–116, 2017.

[9] K. M. Ruff and R. V. Pappu, "Alphafold and implications for intrinsically disordered proteins," *Journal of molecular biology*, vol. 433, no. 20, p. 167208, 2021.

[10] D. Sala, F. Engelberger, H. Mchaourab, and J. Meiler, "Modeling conformational states of proteins with alphafold," *Current Opinion in Structural Biology*, vol. 81, p. 102645, 2023.

[11] G. Janson, G. Valdes-Garcia, L. Heo, and M. Feig, "Direct generation of protein conformational ensembles via machine learning," *Nature Communications*, vol. 14, no. 1, p. 774, 2023.

[12] J. M. Lalmansingh, A. T. Keeley, K. M. Ruff, R. V. Pappu, and A. S. Holehouse, "Soursop: A python package for the analysis of simulations of intrinsically disordered proteins," *Journal of Chemical Theory and Computation*, vol. 19, no. 16, pp. 5609–5620, 2023.

[13] E. Tejada, R. Minghim, and L. G. Nonato, "On improved projection techniques to support visual exploration of multi-dimensional data sets," *Information Visualization*, vol. 2, no. 4, pp. 218–231, 2003.

[14] L. Urry, M. Cain, S. Wasserman, P. Minorsky, and R. Orr, *Campbell Biology*. Pearson, 2020. [Online]. Available: https://books.google.it/books?id=zqFlxwEACAAJ

[15] R. Trivedi and H. A. Nagarajaram, "Intrinsically disordered proteins: an overview," *International journal of molecular sciences*, vol. 23, no. 22, p. 14050, 2022.

[16] M. M. Babu, "The contribution of intrinsically disordered regions to protein function, cellular complexity, and human disease," *Biochemical Society Transactions*, vol. 44, no. 5, pp. 1185–1200, 2016.

[17] P. Tompa, "Intrinsically disordered proteins: a 10-year recap," *Trends in biochemical sciences*, vol. 37, no. 12, pp. 509–516, 2012.

[18] V. N. Uversky, "Intrinsically disordered proteins and their "mysterious"(meta) physics," *Frontiers in Physics*, vol. 7, p. 10, 2019.

[19] ——, "Introduction to intrinsically disordered proteins (idps)," pp. 6557–6560, 2014.

[20] M. Krzeminski, J. A. Marsh, C. Neale, W.-Y. Choy, and J. D. Forman-Kay, "Characterization of disordered proteins with ensemble," *Bioinformatics*, vol. 29, no. 3, pp. 398–399, 2013.

[21] P. Bernadó, E. Mylonas, M. V. Petoukhov, M. Blackledge, and D. I. Svergun, "Structural characterization of flexible proteins using small-angle x-ray scattering," *Journal of the American Chemical Society*, vol. 129, no. 17, pp. 5656–5664, 2007.

[22] L. Salmon, G. Nodet, V. Ozenne, G. Yin, M. R. Jensen, M. Zweckstetter, and M. Blackledge, "Nmr characterization of long-range order in intrinsically disordered proteins," *Journal of the American Chemical Society*, vol. 132, no. 24, pp. 8407–8418, 2010.

[23] S. Rauscher, V. Gapsys, M. J. Gajda, M. Zweckstetter, B. L. De Groot, and H. Grubmüller, "Structural ensembles of intrinsically disordered proteins depend strongly on force field: a comparison to experiment," *Journal of chemical theory and computation*, vol. 11, no. 11, pp. 5513–5524, 2015.

[24] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko *et al.*, "Highly accurate protein structure prediction with alphafold," *nature*, vol. 596, no. 7873, pp. 583–589, 2021.

[25] M. Baek, F. DiMaio, I. Anishchenko, J. Dauparas, S. Ovchinnikov, G. R. Lee, J. Wang, Q. Cong, L. N. Kinch, R. D. Schaeffer *et al.*, "Accurate prediction of protein structures and interactions using a three-track neural network," *Science*, vol. 373, no. 6557, pp. 871–876, 2021.

[26] T. Mittag and J. D. Forman-Kay, "Atomic-level characterization of disordered protein ensembles," *Current opinion in structural biology*, vol. 17, no. 1, pp. 3–14, 2007.

[27] N. Sibille and P. Bernado, "Structural characterization of intrinsically disordered proteins by the combined use of nmr and saxs," *Biochemical society transactions*, vol. 40, no. 5, pp. 955–962, 2012.

[28] H. J. Feldman and C. W. Hogue, "Probabilistic sampling of protein conformations: new hope for brute force?" *Proteins: Structure, Function, and Bioinformatics*, vol. 46, no. 1, pp. 8–23, 2002.

[29] E. Braun, J. Gilmer, H. B. Mayes, D. L. Mobley, J. I. Monroe, S. Prasad, and D. M. Zuckerman, "Best practices for foundations in molecular simulations [article v1.0]," *Living Journal of Computational Molecular Science*, vol. 1, no. 1, p. 5957, Nov. 2018. [Online]. Available: https://livecomsjournal.org/index.php/livecoms/article/view/v1i1e5957

[30] E. Ravera, L. Sgheri, G. Parigi, and C. Luchinat, "A critical assessment of methods to recover information from averaged data," *Physical Chemistry Chemical Physics*, vol. 18, no. 8, pp. 5686–5701, 2016.

[31] M. AlQuraishi, "Machine learning in protein structure prediction," *Current opinion in chemical biology*, vol. 65, pp. 1–8, 2021.

[32] D. Del Alamo, D. Sala, H. S. Mchaourab, and J. Meiler, "Sampling alternative conformational states of transporters and receptors with alphafold2," *Elife*, vol. 11, p. e75751, 2022.

[33] R. A. Stein and H. S. Mchaourab, "Speach_af: Sampling protein ensembles and conformational heterogeneity with alphafold2," *PLOS Computational Biology*, vol. 18, no. 8, p. e1010483, 2022.

[34] H. K. Wayment-Steele, S. Ovchinnikov, L. Colwell, and D. Kern, "Prediction of multiple conformational states by combining sequence clustering with alphafold2," *BioRxiv*, pp. 2022–10, 2022.

[35] PDBe-KB consortium, "Pdbe-kb: collaboratively defining the biological context of structural data," *Nucleic acids research*, vol. 50, no. D1, pp. D534–D542, 2022.

[36] M. Varadi, S. Anyango, M. Deshpande, S. Nair, C. Natassia, G. Yordanova, D. Yuan, O. Stroe, G. Wood, A. Laydon *et al.*, "Alphafold protein structure database: massively expanding the structural coverage of protein-sequence space with high-accuracy models," *Nucleic acids research*, vol. 50, no. D1, pp. D439–D444, 2022.

[37] T. Lazar, E. Martínez-Pérez, F. Quaglia, A. Hatos, L. B. Chemes, J. A. Iserte, N. A. Méndez, N. A. Garrone, T. E. Saldaño, J. Marchetti *et al.*, "Ped in 2021: a major update of the protein ensemble database for intrinsically disordered proteins," *Nucleic acids research*, vol. 49, no. D1, pp. D404–D411, 2021.

[38] A. Hatos, B. Hajdu-Soltész, A. M. Monzon, N. Palopoli, L. Álvarez, B. Aykac-Fas, C. Bassot, G. I. Benítez, M. Bevilacqua, A. Chasapi *et al.*, "Disprot: intrinsic protein disorder annotation in 2020," *Nucleic acids research*, vol. 48, no. D1, pp. D269–D276, 2020.

[39] D. Piovesan, A. Del Conte, D. Clementel, A. M. Monzon, M. Bevilacqua, M. C. Aspromonte, J. A. Iserte, F. E. Orti, C. Marino-Buslje, and S. C. Tosatto, "Mobidb: 10 years of intrinsically disordered proteins," *Nucleic acids research*, vol. 51, no. D1, pp. D438–D444, 2023.

[40] A. Hatos, A. M. Monzon, S. C. Tosatto, D. Piovesan, and M. Fuxreiter, "Fuzdb: a new phase in understanding fuzzy interactions," *Nucleic acids research*, vol. 50, no. D1, pp. D509–D517, 2022.

[41] S. Fukuchi, T. Amemiya, S. Sakamoto, Y. Nobe, K. Hosoda, Y. Kado, S. D. Murakami, R. Koike, H. Hiroaki, and M. Ota, "Ideal in 2014 illustrates interaction networks composed of intrinsically disordered proteins and their binding partners," *Nucleic acids research*, vol. 42, no. D1, pp. D320–D325, 2014.

[42] H. Ghafouri, T. Lazar, A. Del Conte, L. G. Tenorio Ku, P. Tompa, S. C. Tosatto, and A. M. Monzon, "Ped in 2024: improving the community deposition of structural ensembles for intrinsically disordered proteins," *Nucleic acids research*, vol. 52, no. D1, pp. D536–D544, 2024.

[43] Y. Vander Meersche, G. Cretin, A. Gheeraert, J.-C. Gelly, and T. Galochkina, "Atlas: protein flexibility description from atomistic molecular dynamics simulations," *Nucleic acids research*, vol. 52, no. D1, pp. D384–D392, 2024.

[44] R. Brüschweiler, "Efficient rmsd measures for the comparison of two molecular ensembles," *Proteins: Structure, Function, and Bioinformatics*, vol. 50, no. 1, pp. 26–34, 2003.

[45] K. Lindorff-Larsen and J. Ferkinghoff-Borg, "Similarity measures for protein ensembles," *PloS one*, vol. 4, no. 1, p. e4203, 2009.

[46] M. Tiberti, E. Papaleo, T. Bengtsen, W. Boomsma, and K. Lindorff-Larsen, "Encore: software for quantitative ensemble comparison," *PLoS computational biology*, vol. 11, no. 10, p. e1004415, 2015.

[47] R. Appadurai, J. K. Koneru, M. Bonomi, P. Robustelli, and A. Srivastava, "Clustering heterogeneous conformational ensembles of intrinsically disordered proteins with t-distributed stochastic neighbor embedding," *Journal of chemical theory and computation*, vol. 19, no. 14, pp. 4711–4727, 2023.

[48] R. Salomon-Ferrer, D. A. Case, and R. C. Walker, "An overview of the amber biomolecular simulation package," *Wiley Interdisciplinary Reviews: Computational Molecular Science*, vol. 3, no. 2, pp. 198–210, 2013.

[49] A. Vitalis and R. V. Pappu, "Absinth: a new continuum solvation model for simulations of polypeptides in aqueous solutions," *Journal of computational chemistry*, vol. 30, no. 5, pp. 673–699, 2009.

[50] B. R. Brooks, C. L. Brooks III, A. D. Mackerell Jr, L. Nilsson, R. J. Petrella, B. Roux, Y. Won, G. Archontis, C. Bartels, S. Boresch *et al.*, "Charmm: the biomolecular simulation program," *Journal of computational chemistry*, vol. 30, no. 10, pp. 1545–1614, 2009.

[51] K. J. Bowers, E. Chow, H. Xu, R. O. Dror, M. P. Eastwood, B. A. Gregersen, J. L. Klepeis, I. Kolossvary, M. A. Moraes, F. D. Sacerdoti *et al.*, "Scalable algorithms for molecular dynamics simulations on commodity clusters," in *Proceedings of the 2006 ACM/IEEE Conference on Supercomputing*, 2006, pp. 84–es.

[52] M. J. Abraham, T. Murtola, R. Schulz, S. Páll, J. C. Smith, B. Hess, and E. Lindahl, "Gromacs: High performance molecular simulations through multi-level parallelism from laptops to supercomputers," *SoftwareX*, vol. 1, pp. 19–25, 2015.

[53] A. P. Thompson, H. M. Aktulga, R. Berger, D. S. Bolintineanu, W. M. Brown, P. S. Crozier, P. J. In't Veld, A. Kohlmeyer, S. G. Moore, T. D. Nguyen *et al.*, "Lammps-a flexible simulation tool for particle-based materials modeling at the atomic, meso, and continuum scales," *Computer Physics Communications*, vol. 271, p. 108171, 2022.

[54] P. Eastman, J. Swails, J. D. Chodera, R. T. McGibbon, Y. Zhao, K. A. Beauchamp, L.-P. Wang, A. C. Simmonett, M. P. Harrigan, C. D. Stern *et al.*, "Openmm 7: Rapid development of high performance algorithms for molecular dynamics," *PLoS computational biology*, vol. 13, no. 7, p. e1005659, 2017.

[55] J. C. Phillips, D. J. Hardy, J. D. Maia, J. E. Stone, J. V. Ribeiro, R. C. Bernardi, R. Buch, G. Fiorin, J. Hénin, W. Jiang *et al.*, "Scalable molecular dynamics on cpu and gpu architectures with namd," *The Journal of chemical physics*, vol. 153, no. 4, 2020.

[56] R. T. McGibbon, K. A. Beauchamp, M. P. Harrigan, C. Klein, J. M. Swails, C. X. Hernández, C. R. Schwantes, L.-P. Wang, T. J. Lane, and V. S. Pande, "Mdtraj: a modern open library for the analysis of molecular dynamics trajectories," *Biophysical journal*, vol. 109, no. 8, pp. 1528–1532, 2015.

[57] N. Michaud-Agrawal, E. J. Denning, T. B. Woolf, and O. Beckstein, "Mdanalysis: a toolkit for the analysis of molecular dynamics simulations," *Journal of computational chemistry*, vol. 32, no. 10, pp. 2319–2327, 2011.

[58] R. J. Gowers, M. Linke, J. Barnoud, T. J. E. Reddy, M. N. Melo, S. L. Seyler, J. Domanski, D. L. Dotson, S. Buchoux, I. M. Kenney *et al.*, "Mdanalysis: a python package for the rapid analysis of molecular dynamics simulations," Los Alamos National Laboratory (LANL), Los Alamos, NM (United States), Tech. Rep., 2019.

[59] T. D. Romo, N. Leioatts, and A. Grossfield, "Lightweight object oriented structure analysis: tools for building tools to analyze molecular dynamics simulations," *Journal of computational chemistry*, vol. 35, no. 32, pp. 2305–2318, 2014.

[60] M. Münz and P. C. Biggin, "Jgromacs: a java package for analyzing protein simulations," 2012.

[61] M. Biasini, T. Schmidt, S. Bienert, V. Mariani, G. Studer, J. Haas, N. Johner, A. D. Schenk, A. Philippsen, and T. Schwede, "Openstructure: an integrated software framework for computational structural biology," *Acta Crystallographica Section D: Biological Crystallography*, vol. 69, no. 5, pp. 701–709, 2013.

[62] S. O. Yesylevskyy, "Pteros 2.0: Evolution of the fast parallel molecular analysis library for c++ and python," 2015.

[63] D. R. Roe and T. E. Cheatham III, "Ptraj and cpptraj: software for processing and analysis of molecular dynamics trajectory data," *Journal of chemical theory and computation*, vol. 9, no. 7, pp. 3084–3095, 2013.

[64] L. Skjærven, X.-Q. Yao, G. Scarabelli, and B. J. Grant, "Integrating protein structural dynamics and evolutionary analysis with bio3d," *BMC bioinformatics*, vol. 15, pp. 1–11, 2014.

[65] K. Haider, A. Cruz, S. Ramsey, M. K. Gilson, and T. Kurtzman, "Solvation structure and thermodynamic mapping (sstmap): an open-source, flexible package for the analysis of water in molecular dynamics trajectories," *Journal of chemical theory and computation*, vol. 14, no. 1, pp. 418–425, 2018.

[66] J. M. Teixeira, "taurenmd: A command-line interface for analysis of molecular dynamics simulations." *Journal of Open Source Software*, vol. 5, no. 50, p. 2175, 2020.

[67] T. Tubiana, J.-C. Carvaillo, Y. Boulard, and S. Bressanelli, "Ttclust: a versatile molecular simulation trajectory clustering program with graphical summaries," *Journal of chemical information and modeling*, vol. 58, no. 11, pp. 2178–2182, 2018.

[68] L. P. Kagami, G. M. das Neves, L. F. S. M. Timmers, R. A. Caceres, and V. L. Eifler-Lima, "Geo-measures: A pymol plugin for protein structure ensembles analysis," *Computational Biology and Chemistry*, vol. 87, p. 107322, 2020.

[69] C. X. Hernández and V. S. Pande, "Mdentropy: Information-theoretic analyses for molecular dynamics," *Journal of Open Source Software*, vol. 2, no. 19, p. 427, 2017.

[70] A. Lervik, E. Riccardi, and T. S. van Erp, "Pyretis: A well-done, medium-sized python library for rare events," 2017.

[71] D. W. Swenson, J.-H. Prinz, F. Noe, J. D. Chodera, and P. G. Bolhuis, "Openpathsampling: A python framework for path sampling simulations. 1. basics," *Journal of chemical theory and computation*, vol. 15, no. 2, pp. 813–836, 2018.

[72] G. Pérez-Hernández and P. W. Hildebrand, "Mdciao: accessible analysis and visualization of molecular dynamics simulation data," *BioRxiv*, pp. 2022–07, 2022.

[73] B. I. Sejdiu and D. P. Tieleman, "Prolint: a web-based framework for the automated data analysis and visualization of lipid–protein interactions," *Nucleic acids research*, vol. 49, no. W1, pp. W544–W550, 2021.

[74] P. Kunzmann and K. Hamacher, "Biotite: a unifying open source computational biology framework in python," *BMC bioinformatics*, vol. 19, pp. 1–8, 2018.

[75] B. Ellis, J. Stylos, and B. Myers, "The factory pattern in api design: A usability evaluation," in *29th International Conference on Software Engineering (ICSE'07)*. IEEE, 2007, pp. 302–312.

[76] M. Y. Lobanov, N. Bogatyreva, and O. Galzitskaya, "Radius of gyration as an indicator of protein structure compactness," *Molecular Biology*, vol. 42, pp. 623–628, 2008.

[77] Y. Maruyama and A. Mitsutake, "Structural stability analysis of proteins using end-to-end distance: a 3d-rism approach," *J*, vol. 5, no. 1, pp. 114–125, 2022.

[78] V. Nguemaha, S. Qin, and H.-X. Zhou, "Atomistic modeling of intrinsically disordered proteins under polyethylene glycol crowding: quantitative comparison with experimental data and implication of protein–crowder attraction," *The Journal of Physical Chemistry B*, vol. 122, no. 49, pp. 11 262–11 270, 2018.

[79] J. Song, G.-N. Gomes, C. C. Gradinaru, and H. S. Chan, "An adequate account of excluded volume is necessary to infer compactness and asphericity of disordered proteins by forster resonance energy transfer," *The Journal of Physical Chemistry B*, vol. 119, no. 49, pp. 15 191–15 202, 2015.

[80] J. Rudnick and G. Gaspari, "The aspherity of random walks," *Journal of Physics A: Mathematical and General*, vol. 19, no. 4, p. L191, 1986.

[81] B. Lee and F. M. Richards, "The interpretation of protein structures: estimation of static accessibility," *Journal of molecular biology*, vol. 55, no. 3, pp. 379–IN4, 1971.

[82] A. Shrake and J. A. Rupley, "Environment and exposure to solvent of protein atoms. lysozyme and insulin," *Journal of molecular biology*, vol. 79, no. 2, pp. 351–371, 1973.

[83] H. Hofmann, A. Soranno, A. Borgia, K. Gast, D. Nettels, and B. Schuler, "Polymer scaling laws of unfolded and intrinsically disordered proteins quantified with single-molecule spectroscopy," *Proceedings of the National Academy of Sciences*, vol. 109, no. 40, pp. 16 155–16 160, 2012.

[84] G. Tesei, A. I. Trolle, N. Jonsson, J. Betz, F. E. Knudsen, F. Pesce, K. E. Johansson, and K. Lindorff-Larsen, "Conformational ensembles of the human intrinsically disordered proteome," *Nature*, vol. 626, no. 8000, pp. 897–904, 2024.

[85] M. J. Pietal, J. M. Bujnicki, and L. P. Kozlowski, "Gdfuzz3d: a method for protein 3d structure reconstruction from contact maps, based on a non-euclidean distance function," *Bioinformatics*, vol. 31, no. 21, pp. 3499–3505, 2015.

[86] M. M. Flocco and S. L. Mowbray, "Cα-based torsion angles: A simple tool to analyze protein conformational changes," *Protein Science*, vol. 4, no. 10, pp. 2118–2122, 1995.

[87] W. Kabsch and C. Sander, "Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features," *Biopolymers: Original Research on Biomolecules*, vol. 22, no. 12, pp. 2577–2637, 1983.

[88] G. Jeschke and L. Esteban-Hofer, "Integrative ensemble modeling of proteins and their complexes with distance distribution restraints," in *Methods in Enzymology*. Elsevier, 2022, vol. 666, pp. 145–169.

[89] G. Jeschke, "Protein ensemble modeling and analysis with mmmx," *Protein Science*, vol. 33, no. 3, p. e4906, 2024.

[90] J. Yang, I. Anishchenko, H. Park, Z. Peng, S. Ovchinnikov, and D. Baker, "Improved protein structure prediction using predicted interresidue orientations," *Proceedings of the National Academy of Sciences*, vol. 117, no. 3, pp. 1496–1503, 2020.

[91] G. Janson and M. Feig, "Transferable deep generative modeling of intrinsically disordered protein conformations," *PLOS Computational Biology*, vol. 20, no. 5, p. e1012144, 2024.

[92] D. Rosenbaum, M. Garnelo, M. Zielinski, C. Beattie, E. Clancy, A. Huber, P. Kohli, A. W. Senior, J. Jumper, C. Doersch *et al.*, "Inferring a continuous distribution of atom coordinates from cryo-em images using vaes. arxiv," *Preprint arXiv*, vol. 210614108, 2021.

[93] F. Nielsen, "On the jensen–shannon symmetrization of distances relying on abstract means," *Entropy*, vol. 21, no. 5, p. 485, 2019.

[94] Y. Rubner, C. Tomasi, and L. J. Guibas, "A metric for distributions with applications to image databases," in *Sixth international conference on computer vision (IEEE Cat. No. 98CH36271)*. IEEE, 1998, pp. 59–66.

[95] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne." *Journal of machine learning research*, vol. 9, no. 11, 2008.

[96] S. Arora, W. Hu, and P. K. Kothari, "An analysis of the t-sne algorithm for data visualization," in *Conference on learning theory*. PMLR, 2018, pp. 1455–1462.

[97] H. Abdi and L. J. Williams, "Principal component analysis," *Wiley interdisciplinary reviews: computational statistics*, vol. 2, no. 4, pp. 433–459, 2010.

[98] B. Schölkopf, A. Smola, and K.-R. Müller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural computation*, vol. 10, no. 5, pp. 1299–1319, 1998.

[99] Q. Luo, L. Christino, F. V. Paulovich, and E. Milios, "Dimenfix: A novel meta-dimensionality reduction method for feature preservation," *arXiv preprint arXiv:2211.16752*, 2022.

[100] L. McInnes, J. Healy, and J. Melville, "Umap: Uniform manifold approximation and projection for dimension reduction," *arXiv preprint arXiv:1802.03426*, 2018.

[101] J. González-Delgado, P. Bernadó, P. Neuvial, and J. Cortés, "WARIO: Weighted families of contact maps to characterize conformational ensembles of (highly-)flexible proteins." [Online]. Available: https://www.researchsquare.com/article/rs-4149901/v1

[102] J. Lincoff, M. Haghighatlari, M. Krzeminski, J. M. Teixeira, G.-N. W. Gomes, C. C. Gradinaru, J. D. Forman-Kay, and T. Head-Gordon, "Extended experimental inferential structure determination method in determining the structural ensembles of disordered protein states," *Communications chemistry*, vol. 3, no. 1, p. 74, 2020.

[103] D. H. Brookes and T. Head-Gordon, "Experimental inferential structure determination of ensembles for intrinsically disordered proteins," *Journal of the American Chemical Society*, vol. 138, no. 13, pp. 4530–4538, 2016.

Acknowledgments

I would like to express my deepest gratitude to all the members of the BioComputing UP lab for their invaluable support and encouragement throughout this project. I am especially thankful to Prof. Alexander Miguel Monzon, my thesis advisor, for his expert guidance at every stage of this journey. My sincere thanks also go to Prof. Silvio Tossato for the opportunity to complete an internship and collaborate on my thesis with some of the best colleagues I've had the pleasure of working with. I am particularly grateful to Hamidreza Ghafouri, with whom I closely collaborated on the development of IDPET and who has been an immense support. Additionally, I wish to thank all the professors from the Erasmus Mundus Big Data Management and Analytics (BDMA) consortium, especially Prof. Esteban Zimanyi and Prof. Massimiliano de Leoni, for providing a uniquely enriching experience throughout the joint master's degree program. Lastly, I am profoundly grateful to my family and friends for always believing in me.