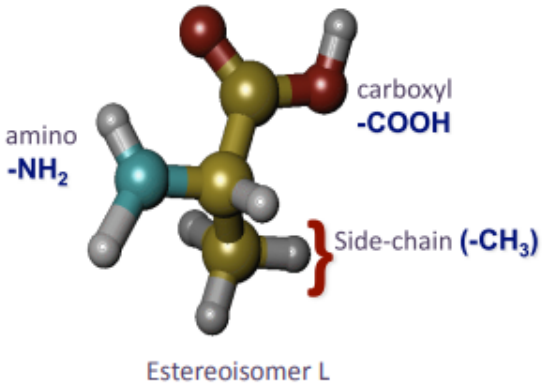
Principles of Protein Structure and Conformational Space

Introduction. Amino Acids

Proteins are polymers. In other words, they are a chemical structure formed by a component that is repeated many times (in this case, amino acids).



Aa have the following structure:

- Carboxyl group
- Amino group
- Alpha carbon
- Side chain

The alpha carbon has an hybridization Sp3 (this is why it has a shape of a tetramer).
Look at the difference between stereoisomers (change position of H and -NH2).

Amino acids are L stereoisomers (we can produce artificially D stereoisomers, but they don't exist in real life).

The important thing is the side chain because it gives the properties to the aa:

- **Polar:** They have an alcohol group (**serine** and **threonine**), sulfur (**cysteine**), amino group (**asparagine** and **glutamine**), all the **charged aa** and **proline** .

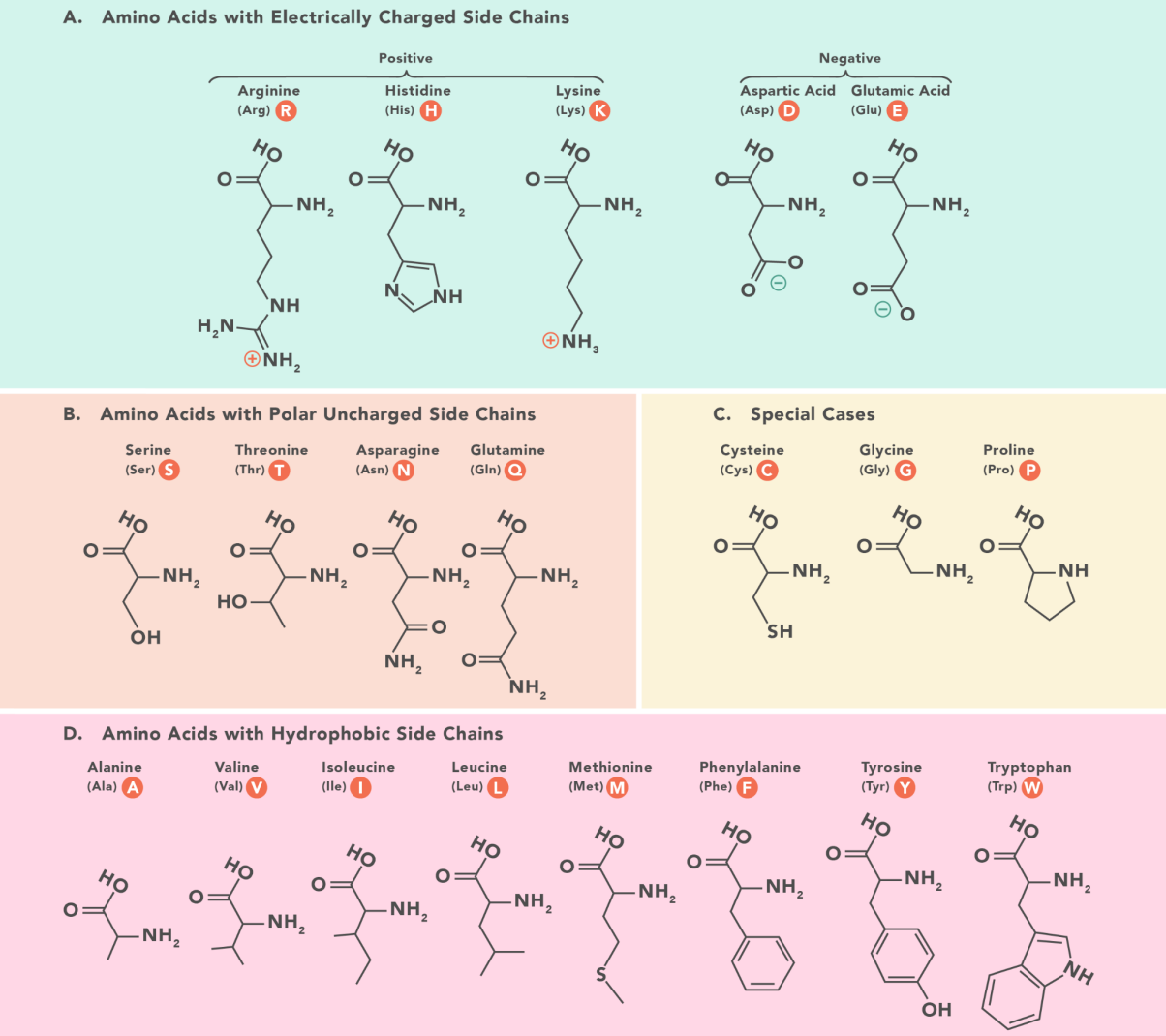
- **Apolar:** Glycine, alanine, valine…

To detect if an aa is polar or apolar, we need to perform an experiment in which we calculate the partition of the aa in water and octanol. We just look at the concentration of that aa in both mediums.

Proline has a ring.

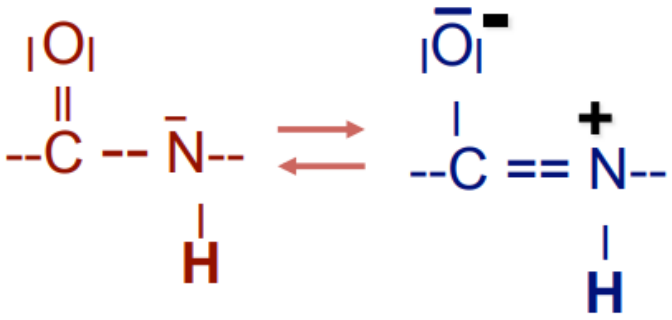
There are also Aromatic groups that are characterized by having benzene (Phenylalanine, Tyrosine and Tryptophan). All the carbones of that ring are in a Sp2 hybridization so they share the P orbitals.
These kind of rings can interact between them and they make something called "**stacking interaction**" (also found in the DNA). This is a key interaction.

There are Aa that are positive (Lysine, Arginine and Histidine, in ph = 7) or negatively charged (Aspartate and Glutamate).



Introduction. Peptide Bond

To make a polymer, we need to join several aa. To join them, there is a condensation reaction. From the carboxylic group of an Aa, the hydrogen goes to the amino group of another Aa and there is a formation of a double bond between the nitrogen and the carboxyl carbon.

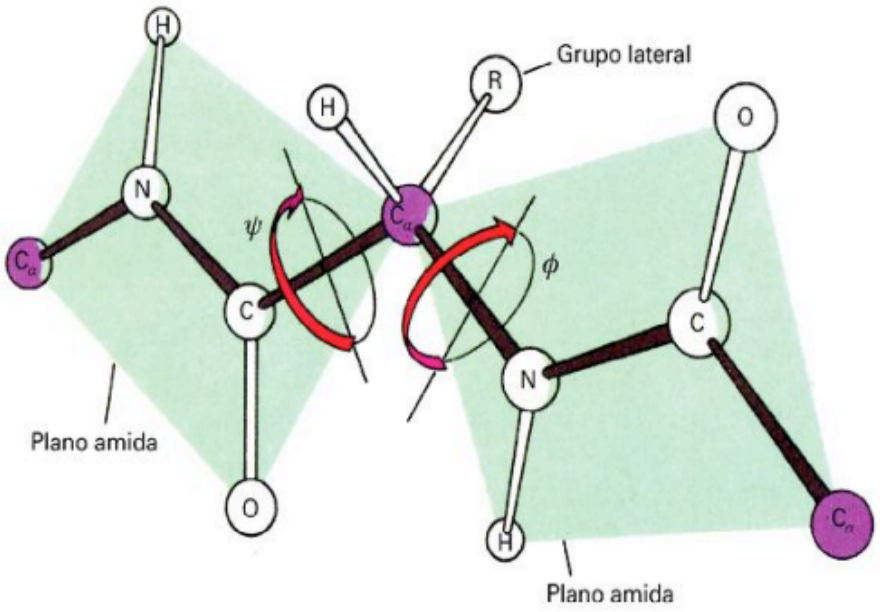


As a consequence a molecule of water is removed.

This double bond can not rotate. Thus, the peptide bond looks like a plane which gives some properties that are special.

Therefore, the only way we can rotate the angles is around the alpha C:

- Phi and Psi angles



Introduction. Native Conformation

If we start rotating these angles, I will obtain a new conformation for my protein.
At some point, there is a conformation that is highly stable and therefore they "survive" for some time. At this stable conformation, the protein can be active (the protein will have a function). This is what we call the "native structure of the protein".

This does not mean that it is the most stable one. It's just stable enough to survive some time in water and perform its function.

- So it highly depends on the polypeptide and environment!

Introduction. Levels of structure

We have seen that the sequence (primary structure) takes a conformation in the space (secondary structure) and some of these structures are highly regular:

- Alpha helix
- Beta strands

The combination of many secondary structures produces the tertiary structure (one single peptide). The combination of a low number of secondary structures produces a "super-secondary" structure.

Some of these tertiary structures interact between them and form the quaternary structures (multiple peptides interacting).

Secondary Structure. Bonding energy terms

If we want to understand this system, we need to look at the energies:

- Bonding energies
- Non-bonding energies

$$V = \frac{1}{2}\sum_i K_i^{bond}\left(d_i - d_{0,i}\right)^2 + \frac{1}{2}\sum_i K_i^{angle}\left(\alpha_i - \alpha_{0,i}\right)^2 + \frac{1}{2}\sum_i K_i^{dihedral}\left(\omega_i - \omega_{0,i}\right)^2$$

$$+\frac{1}{2}\sum_i K_i^{torsion}\cos\left(\lambda_i\phi_i + \delta_i\right) + \frac{1}{4\pi\varepsilon}\sum_i\sum_{j>i}\frac{q_i q_j}{r_{ij}} + \sum_i\sum_{j>i}\left(\frac{C_6(i,j)}{r_{ij}^6} - \frac{C_{12}(i,j)}{r_{ij}^{12}}\right)$$

$$+\sum_i\left(Hydrogen - Bonds\right) + \sum_i\left(\pi - \pi\right)$$

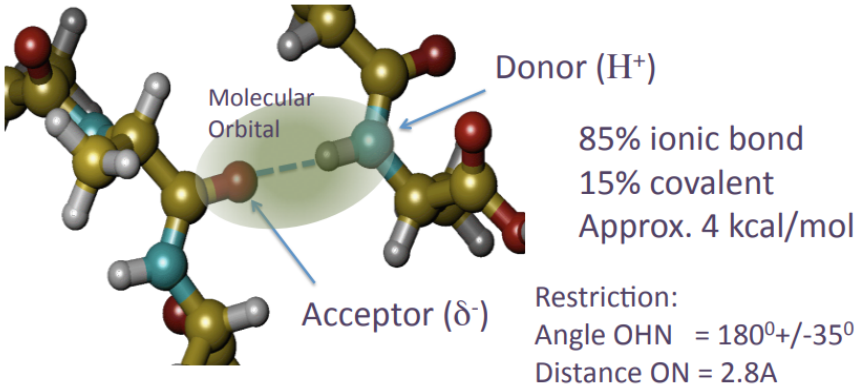
Secondary Structure. Hydrogen Bonds

The Hydrogen bonds are produced by one hydrogen getting highly positive (from the amino group) and one atom that has a tendency to be negatively charged (Oxigen of a carbonyl group, for example).

This happens to be in all peptide bonds, because you have the NH group and the COOH group. So, we can have these kinds of interactions between hydrogens and oxygens along the sequence of the protein.

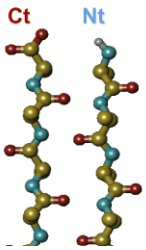
The Oxygen with the Hydrogen and Nitrogen makes a single molecular orbital. This means that there is a straight bond that makes the O, H and N look like they are in a single line:

- Angle OHN = 180º +- 35º
- Distance = 2.8A
- Energy = Approx 4 Kcal/mol
    - 85% ionic bond
    - 15% covalent



They are very important, not because of their energy but because there are a lot. This makes it look like a zipper.

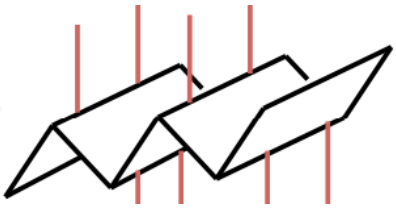
Secondary Structure. Strands



Residue 1 makes a hydrogen bond with residue 134, residue 2 with 133… for example.

- Hydrogen bonds are long distances in sequence.

This is a beta ladder.
If we join this with a third strand, then it will look like a sheet (beta sheet).



There is a need to make some twists in the psi and phi angles to make the hydrogen bonds. At the end, the side chains go towards one side or the other

Lets imagine that the sequence is:
Polar = P
Non-Polar = N

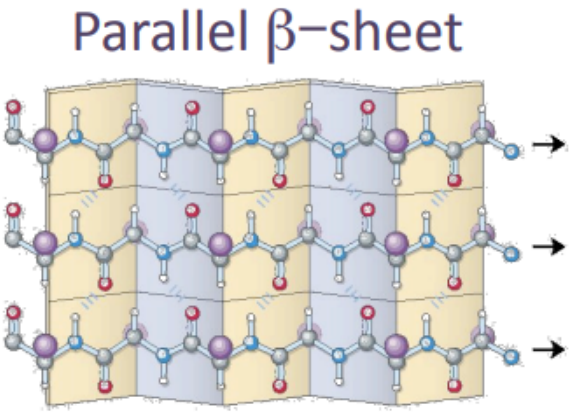
P-N-P-N-P-N-P-N

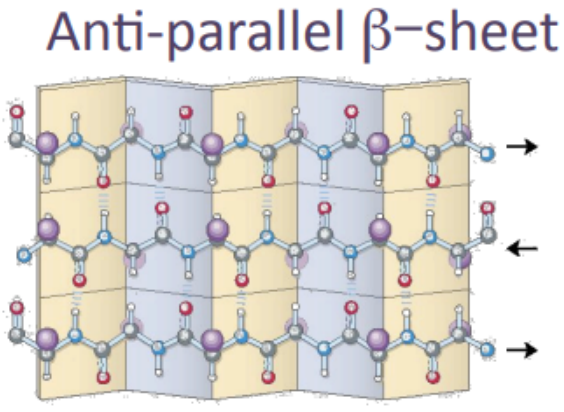
This will make one of the sides of the beta sheet polar and the other side non-polar.

Note that the beta ladder is antiparallel (one goes up and the other one goes down).

If they are parallel:

- Residue 1 will interact with residue 101, residue 2 with 102…



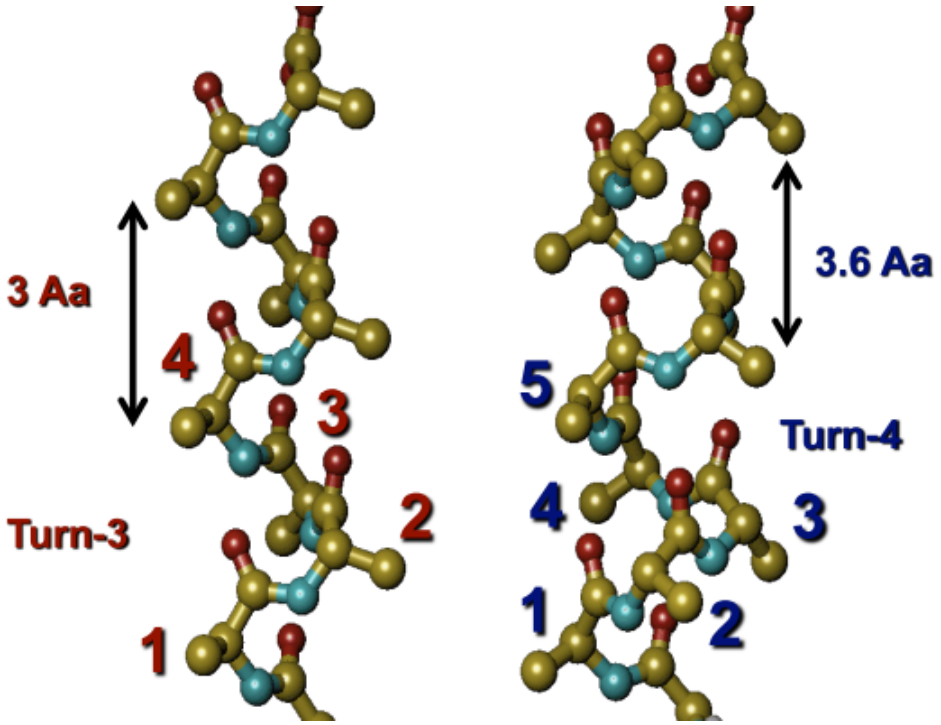


Secondary Structure. Helices

In this case, the hydrogen bonds are not that far separated along the sequence.

- Residue 1 interacts with residue 4 (not closer aa otherwise it is not possible). There is a distance of 3 Aa → Helix 3-10
- If there is a distance of 4 Aa, then we make an alpha helix (most stable).
- There is no distance of 5 Aa because the helix would be so wide that the water would break the hydrogen bonds.

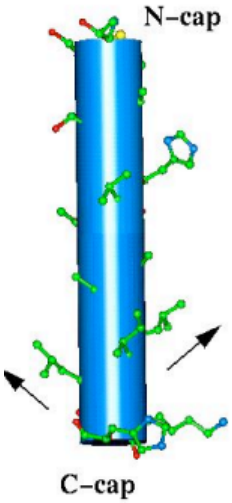
Note that the side chains protrude out of the helix.



Helixes are not infinite long. At some point, you have to end them.
The thing is that you will lose some of the hydrogen bonds and therefore you need to stabilize the structure somehow.

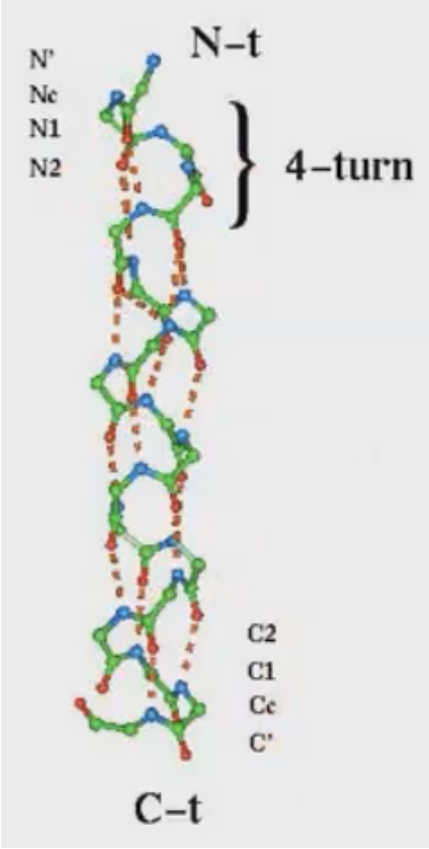
- It gets stabilized by the regions called “cap”.

So, it must get stabilized by some change in the hydrogen bonding network and some interactions between the last residues.



Depending on how we break the ends, we will obtain different types:

	Hbond	Hidrofobic
N-cap box	N_c-N_3	N'-N_4
Big box	N_c-N_3	N''-N_4
Shellman	C''-C_3 C'-C_2	C''-C_3
α_L	C'-C_3	C''-C_3



N'' is on top of N' and C'' is below C'

The hydrophobic interactions substitute the hydrogen bonds.

If we look at the image, we can see that the carboxyles are looking down and the nitrogens look up. This produces a dipole and therefore the helix will have some dipole:

- N-cap will be positively charged
- C-cap will be negatively charged

If we want to stabilize the helix, we will want to:

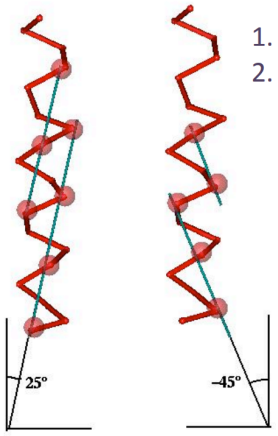
- Add residues that are negatively charged at the N-cap
- Add residues that are positively charged at the C-cap

Because this will neutralize the dipole.

Moreover, the helices have faces. We can see a region that is polar and another one that is non-polar (the helix is amphipathic).

We can also look at how the side chains are placed with respect to the axis of the helix:

- If we look from left to right, there is an angle of 25º
- If we look from right to left, there is an angle of -45º



Secondary Structure. Ramachandran Plot

We have seen that we are going to place the aa in such a conformation that they make hydrogen bonds that lead to the regular secondary structures.
They are "regular" because they have been seen very often.

If we plot a 2 dimensional diagram:

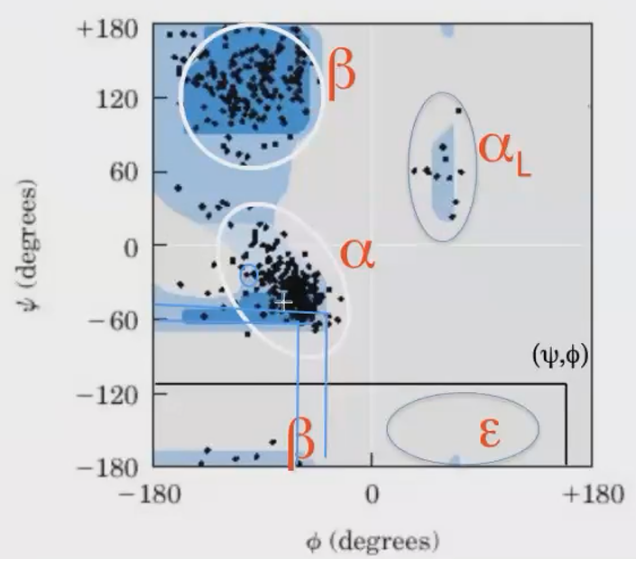
- X axis: PHI
- Y axis: PSI

Each dot corresponds to the angles an aa has in a certain protein.
There are some angles that are more common in helixes and others in sheets.
Note that not every combination of angles is stable.

At the end, all the regions that are regular have clustered all the dots.

The dots outside the areas are not seen very often and it is a little bit weird.

Each protein has its own Ramachandran plot



Secondary Structure. Aa propensity

Certain aa occur very often with specific secondary structures. For example:

- Glu, Met, Ala appear very often in alpha helices.
- Phe, Ile, Val appear very often in beta sheets.
- Pro and Gly appear very often in the turns (not common in alpha helices or beta sheets). Because:
    - Prolines do not make Hydrogen bonds and therefore they can not appear in regular secondary structure otherwise it would break the structure.

The first methods of prediction use the peptide sequence to predict how it is going to fold.

Soluble and Globular proteins

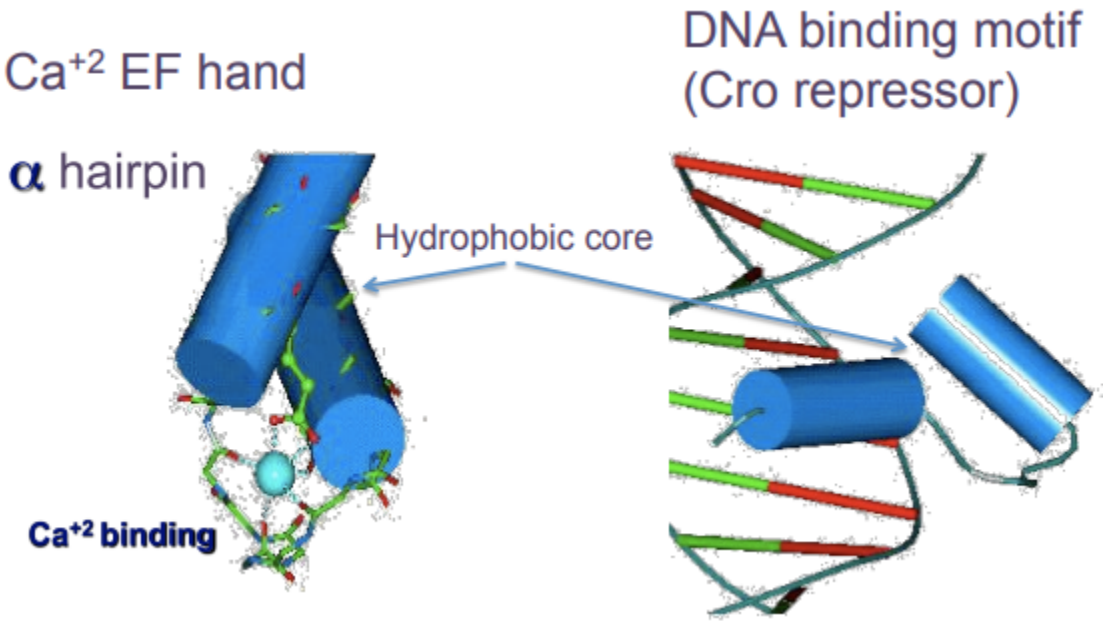
Motif or Super-Secondary Structure: Cluster of 2-4 regular secondary structures, usually involved in a particular function.

- Regular secondary structures are connected by **loops**.
- Loops are highly flexible and produce change of the polypeptide chain orientation.
- Length of loops varies from 1 to more than 40 residues
- Some super-secondary structures are stabilized with an hydrophobic core

Supersecondary structure. Alpha - Alpha

For example:

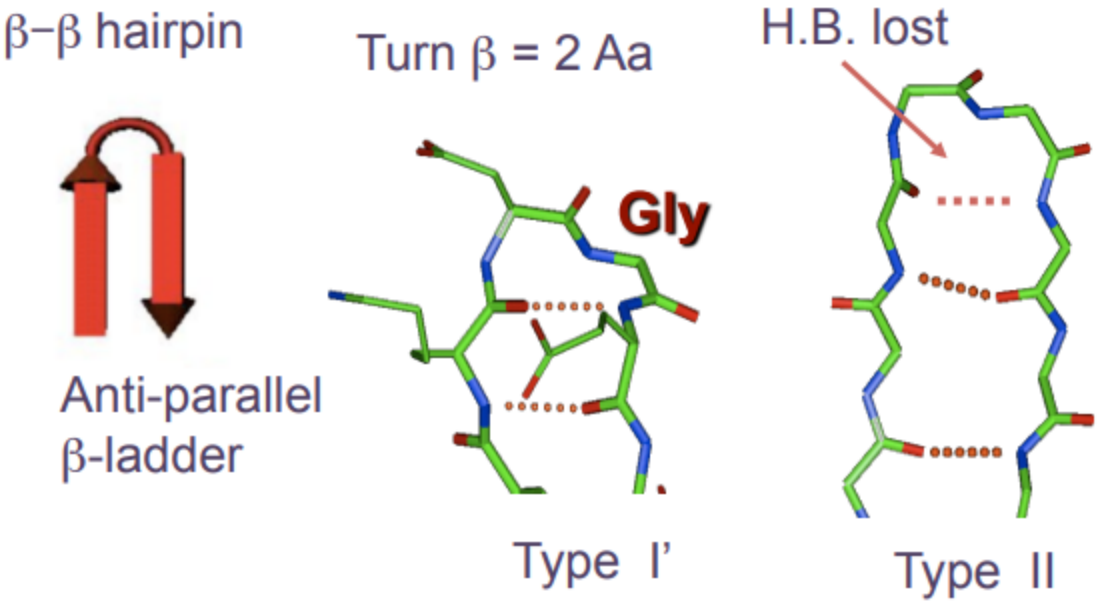
- Alpha hairpin
- DNA binding motif



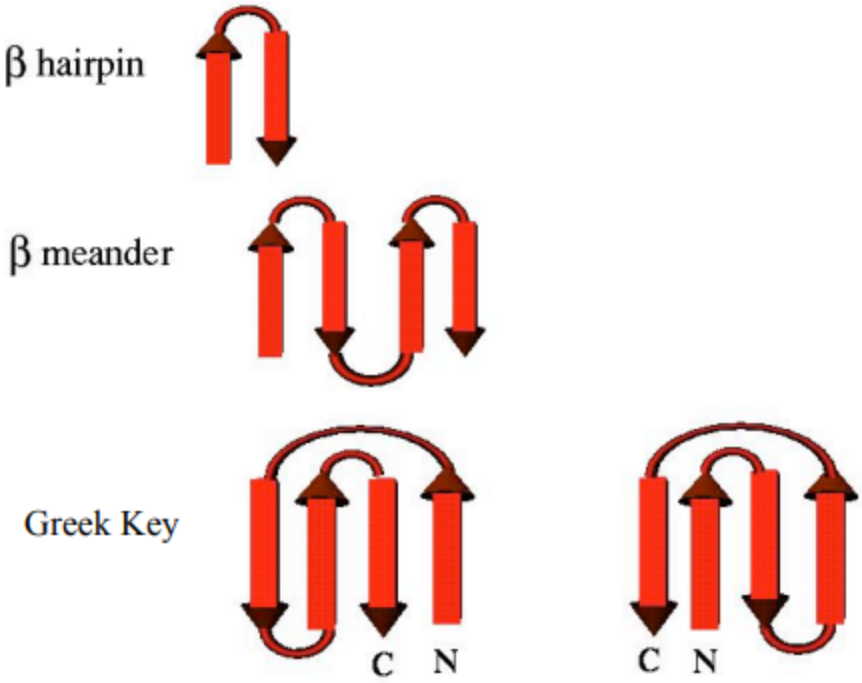
Supersecondary structure. Beta - Beta

For example:

- Anti parallel beta-ladder



We can join more than one beta hairpin

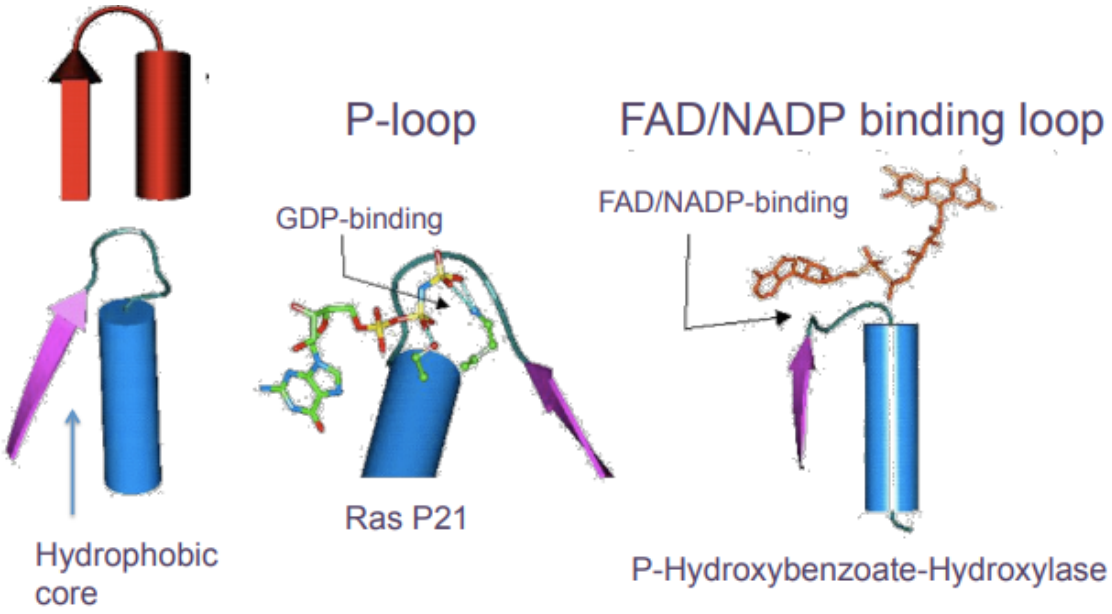


Look that both Greek keys are different. If you superpose one over the other, you will see that they are different structures.

Supersecondary structure. Beta - Alpha & Alpha - Beta

For example:

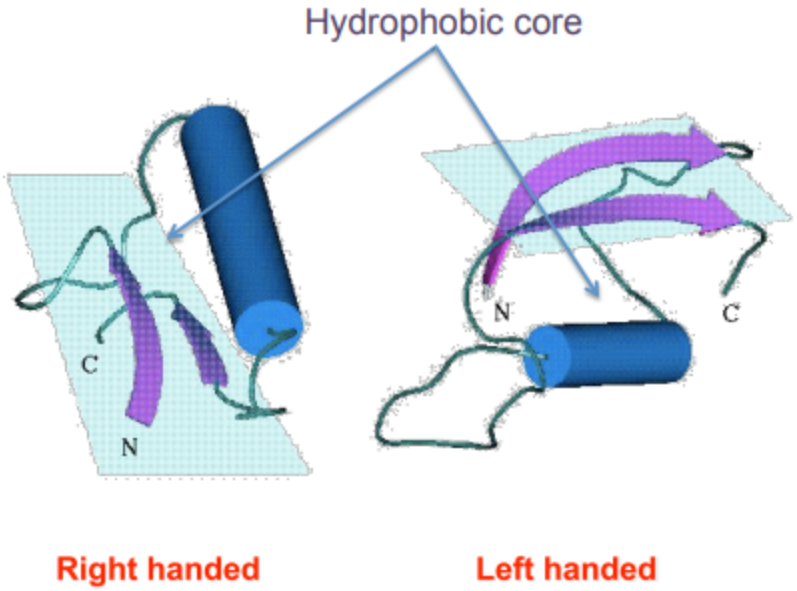
- P-loop



Supersecondary structure. Beta - Alpha - Beta

There are 2 types:

- Right handed
- Left handed



Domains

The protein domain is defined as the fundamental unit of 3D structure, able to fold by itself in the right conditions with independence of the rest of the protein. It often corresponds to functional local and compact units of a protein.

They are able to join several regular secondary structures to form a fold.

Classification

One protein can be constructed by several domains and each domain can be constructed by several super-secondary structures and each secondary structure is constructed by 2-4 regular secondary structures.

There are:

- All alpha type domains. There are only alpha helices.
- All beta type domains. There are only beta strands.
- Alpha / Beta domains. Alpha, beta, alpha, beta…
- Alpha + Beta domains. There is one region with alpha helices and one region with beta strands.

In between of this types, you have different types of domains (not important)

Function - Sequence - Structure

In order to classify and give some order to all these types of folds, we are also going to analyze the sequence and the function.

Even though two sequences are different, the structure can still be conserved and thus they may perform a similar function.

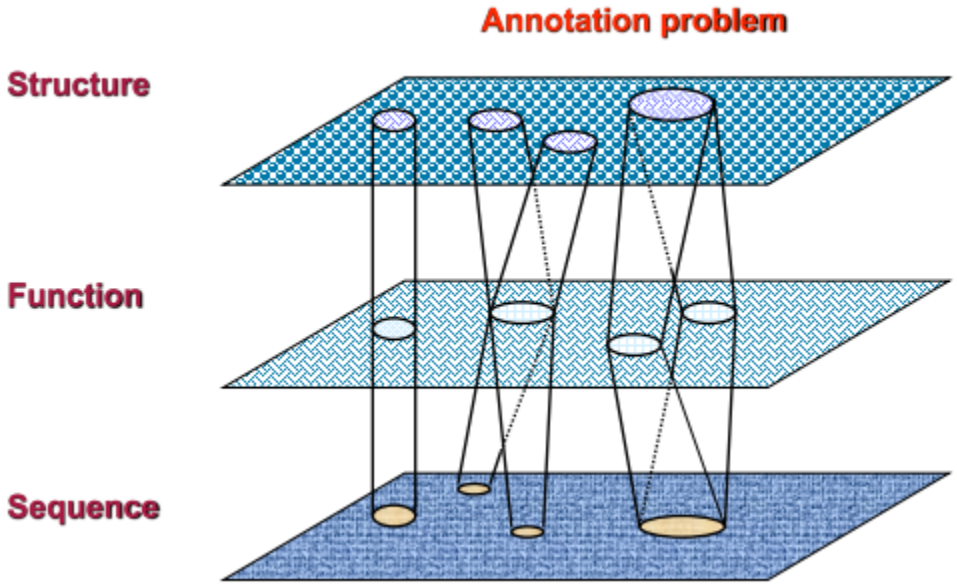
Homology: Proteins are homologous if they are the products of genes that evolved from the same ancestor.

There are proteins that have a very similar sequence, a very similar function and a very similar structure. They are **orthologs (**speciation**)** and they are members of the same family.

There are proteins that have different sequences, different functions but the same structure. They have converged and they are **analogs**. NO COMMON ANCESTOR

There are proteins that have different sequences, the same function and structure. Probably because evolution has changed the sequences so much, but they are still homologs. There is a **remote homology** and we say that they are members of the same superfamily.

Finally, there are sequences with very similar sequences and structures but a different function. They are **paralogs**. A gene that has duplicated and has had a lot of mutations that have led to a new function.



SCOP DB

The classification of structures that we have seen is the classical one for the SCOP classification. They have been classified by:

- **Class:** It groups the folds according to the percentage and 3D disposition of the regular secondary structures. All alpha, all beta…
- **Family:** It groups proteins clearly related by homology. In general we assume they are homologs if the alignment has >30% ID, they have the similar structure and similar function
- **Superfamily:** Proteins which sequences align with very few %ID but showing similar structural patterns and similar function. Remote homology
- **Fold:** Proteins with very similar disposition of the regular secondary structures

It is done manually

CATH DB (done automatically)

Class(C)
Class is determined according to the secondary structure composition and packing within the structure. It is assigned automatically

Architecture(A)
This describes the overall shape of the domain structure as determined by the orientations of the secondary structures but ignores the connectivity between the secondary structures. It is currently assigned **manually**

Topology(T)
Structures are grouped into fold groups at this level depending on both the overall shape and connectivity of the secondary structures. This is done using an automated structure comparison algorithm.

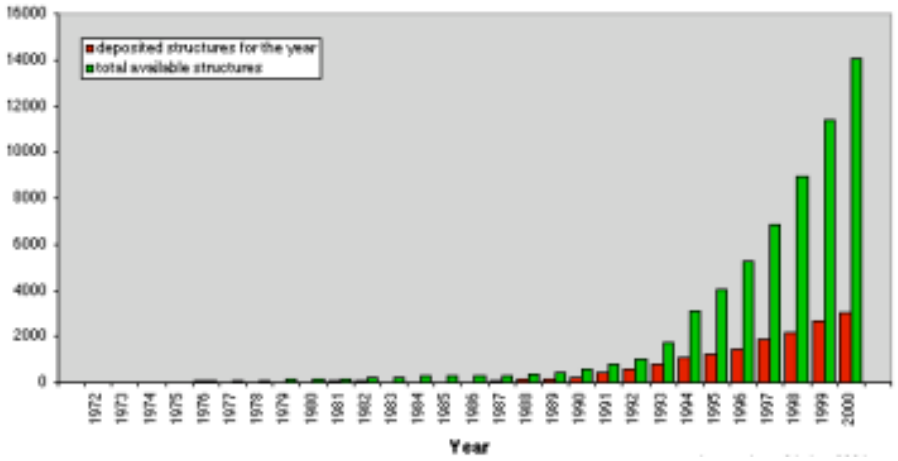
Homologous superfamily (H)
This level groups together protein domains which are thought to share a common ancestor and can therefore be described as homologous. Similarities are automatically identified either by high sequence identity or structure comparison.

Sequence Family Levels: (S,O,L,I, D)
Domains within each H-level are subclustered into sequence families using multi-linkage clustering S(35%), O(60%, L (90%), I (100%)

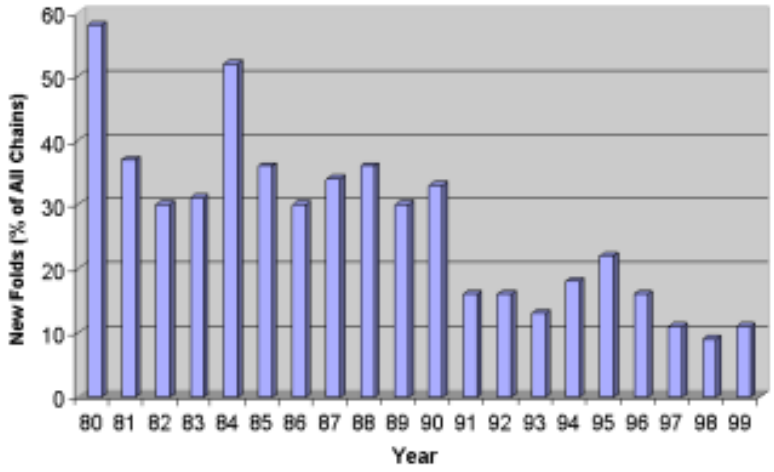
Protein Structure

Soluble and globular proteins. Principles observed in folds

There has been a geometric increase of the number of structures in the PDB.



The % of new folds has reached a plateau and the tendency will be to decrease until no more folds are found. This is very important because it means that the number of folds is limited, meaning that with the tools we have today we can predict the folds. If the number of folds was infinite, we would not have the capability of predicting.



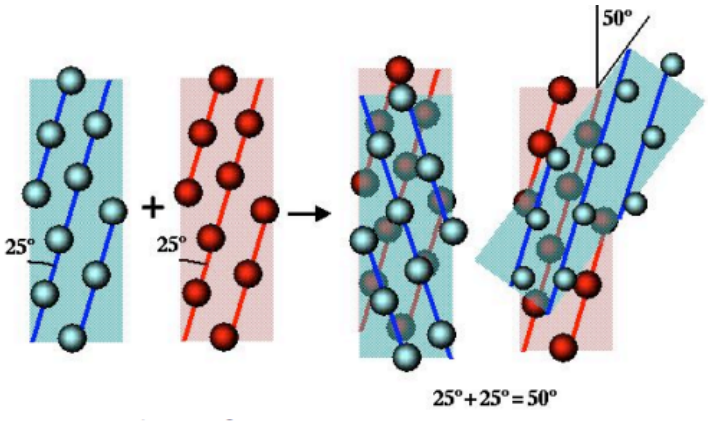
1. Folds have a compact structure formed by the organization of regular secondary structures.
2. Folds have an hydrophobic core that stabilizes its conformation.
3. The hydrophobic core is formed by non-polar residues, while polar residues are mostly in the surface and active sites.
4. Structure is conserved among proteins with similar sequences.
5. The total number of folds is finite. With the knowledge we have now a days, we can predict the fold of other proteins from which we do not know its structure.
6. The structure of a protein can be built by joining several folds (this can be produced by gene fusion).

Soluble and globular proteins. All alpha

Tenen un core hidrofòbic fet per les zones hidrofòbiques de les cadenes alfa empaquetades a dins i les zones hidrofíliques a fora. Les cadenes han de formar un angle de 50º (25+25) o 20º (45-25).

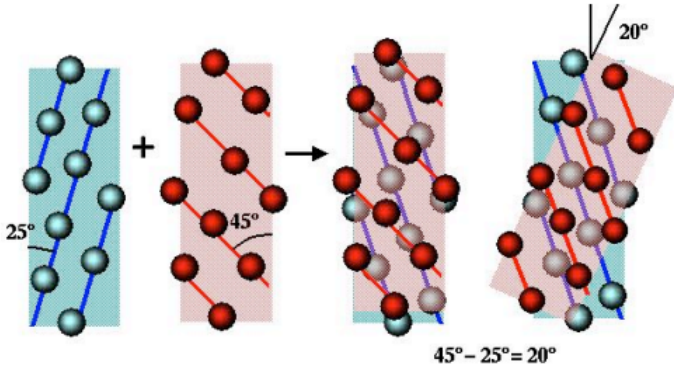
- 4 helix bundle: 4 hèlixs alfa amb un angle de 20º formant dues parelles, unit per loops.
- Globin like: 8 hèlixs alfa connectades per loops curts i amb un nucli hidrofòbic. Trobem els dos tipus d'angle

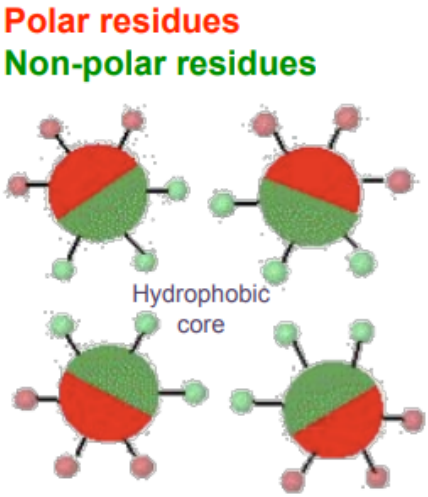
As we remember, the angle that forms an alpha helix is equal to 25º. We just have to join 2 helices in different directions.
As a result, they form 50º.



If we change our point of view, the alpha helices also have a 45º angle.

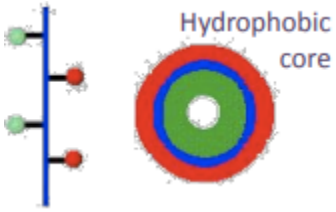
Some alpha helices are amphipathic and therefore, they can build an hydrophobic core.





Soluble and globular proteins. All beta

In beta strands, the residues are alternatively polar and non-polar. This allows the creation of a hydrophobic core.



Beta barrel: It consists of a cylindrical structure made up of multiple beta-strands arranged in a parallel or antiparallel fashion. Hydrophobic core and exterior face is non-polar.

Beta propeller or superbarrel: Formed by 6 beta meanders placed as blades (both faces are hydrophilic)

Greek key Beta-sandwich: 2 Greek keys, one in front of the other. Both have a polar and non-polar face.

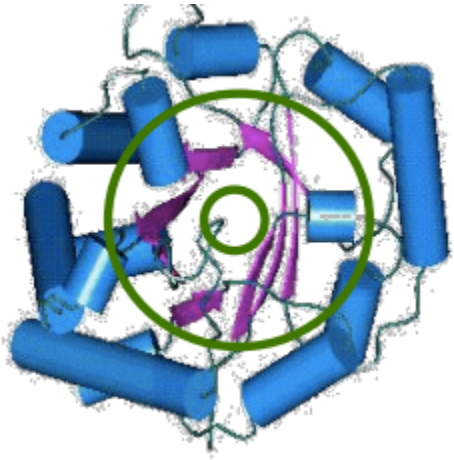
Jelly-Roll: It consists of two beta-sheets, each made up of four antiparallel beta-strands, arranged in a "jellyroll" or "sandwich" configuration. The two sheets are arranged in a "Greek key" topology.

Immunoglobulin-like: It consists of a beta-sandwich structure made up of two beta-sheets, each consisting of several antiparallel beta-strands. The two sheets are arranged in a "Greek key" topology.

Soluble and globular proteins. Alpha / Beta

The TIM barrel has the following construction:

- Beta Strands forming a cylinder
- Alpha helices surrounding the beta strands.



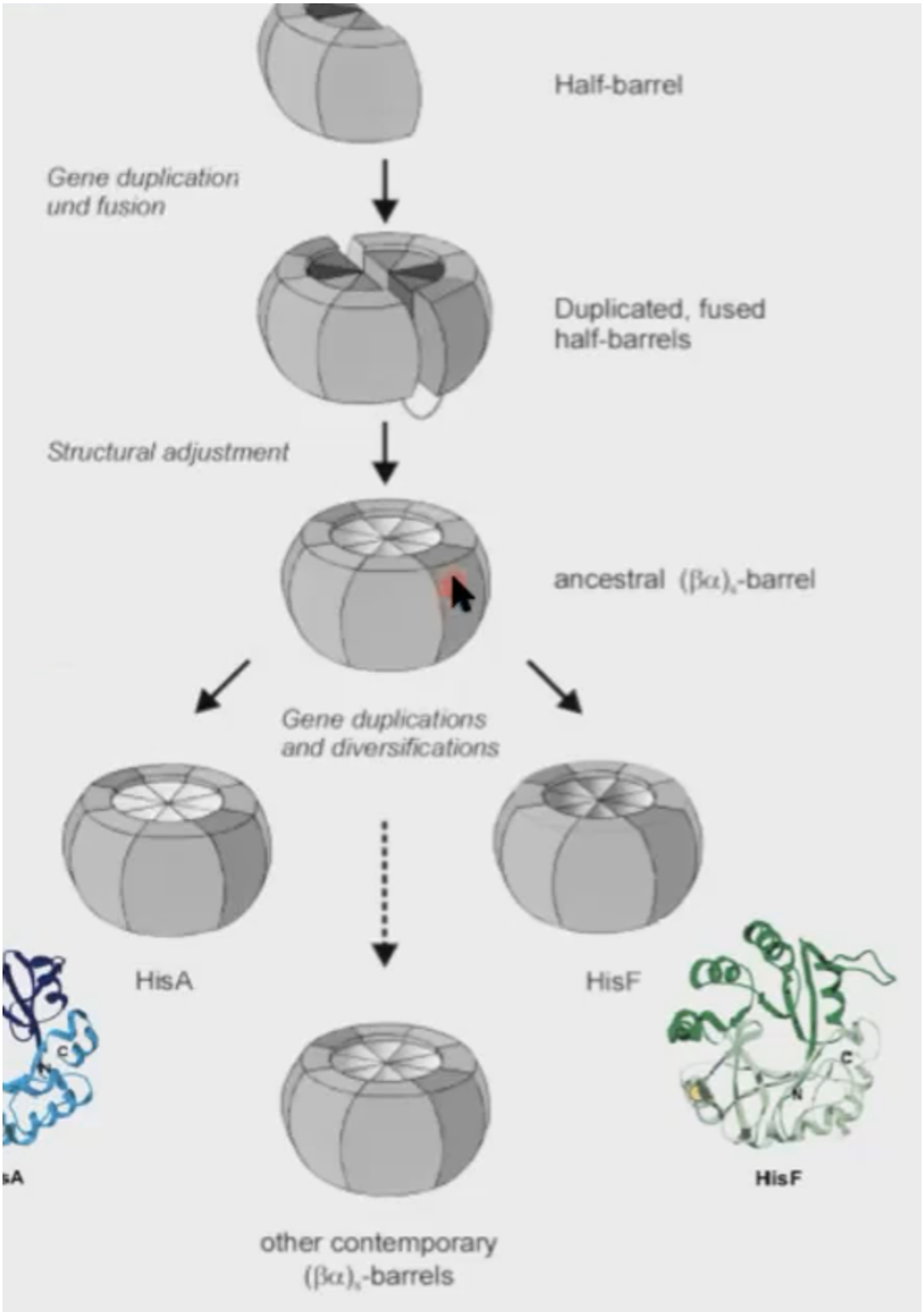
It is highly stable due to the double hydrophobic core:

- Core 1: Inside the beta barrel
- Core 2: Between the alpha helices and the beta strands

It is found in many proteins, mostly enzymes. Even though their sequences are really different, their structures are really similar (analogs).

Actually, it has been proved that all of them have the same ancestor.

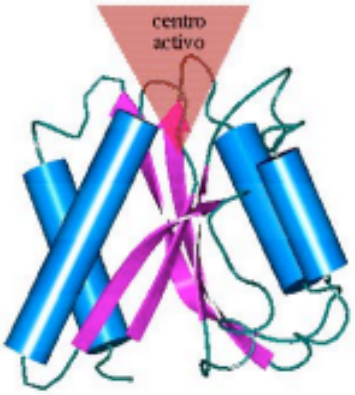
The TIM barrel comes from a gene duplication, followed by a gene fusion.

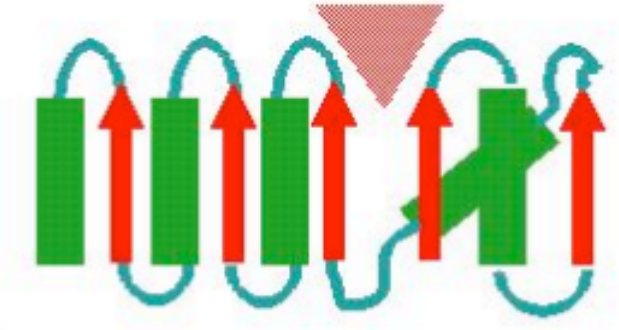


Soluble and globular proteins. Rossmann Fold

The Rossmann fold is a beta sheet surrounded by alpha helices.

It is often found as a binding domain of mono-nucleotides: ATP, ADP, NAD, FAD…
So, enzymes that require some energy (ATP) will incorporate this fold.

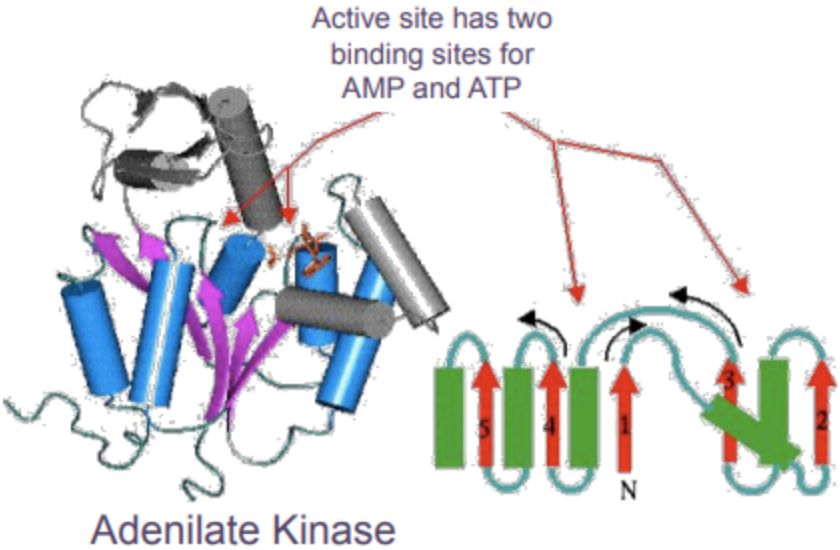




It is possible to predict where is the active site knowing the structure:

- The active site is located in a Beta-Alpha motif of the type NADP/FAD binding loop
- The location of the loop was often found in the change of orientation of the **topological diagram**. This is called a **topological switch point.**

Topological diagram is the picture on the right. It's just putting in a 2D plane the structure.



In the Ct of the β strand of the β-α loop where it changes the orientation of the topological diagram.

Globular membrane proteins. Definition

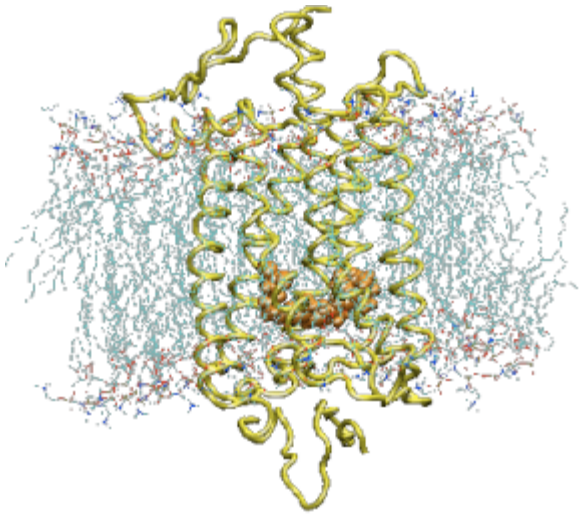
- Membrane Proteins can be of two types: peripheral and integral
    - Integral membrane proteins are inserted in the membrane, where the main bulk of the domain is buried inside the membrane
    - Peripheral membrane proteins are only linked by few residues to the membrane

- The residues of the integral proteins are mostly non-polar, changing the rules of hydrophobicity patterns described for soluble domains.

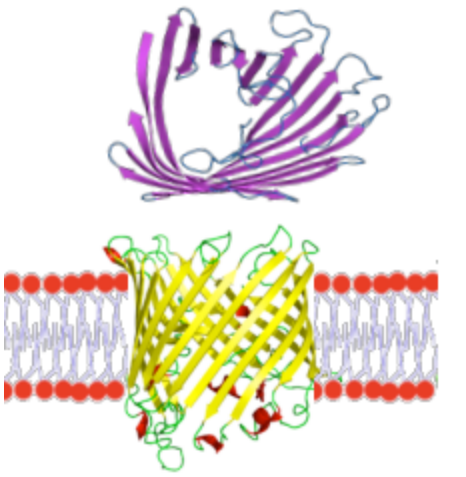
Globular membrane proteins. Integral

Two types of integral proteins:

- α-helix transmembrane (these helices are hydrophobic, reason why they are inserted in the membrane). The loops that are outside the membrane are hydrophilic.
- β-sheet transmembrane



α-helix transmembrane



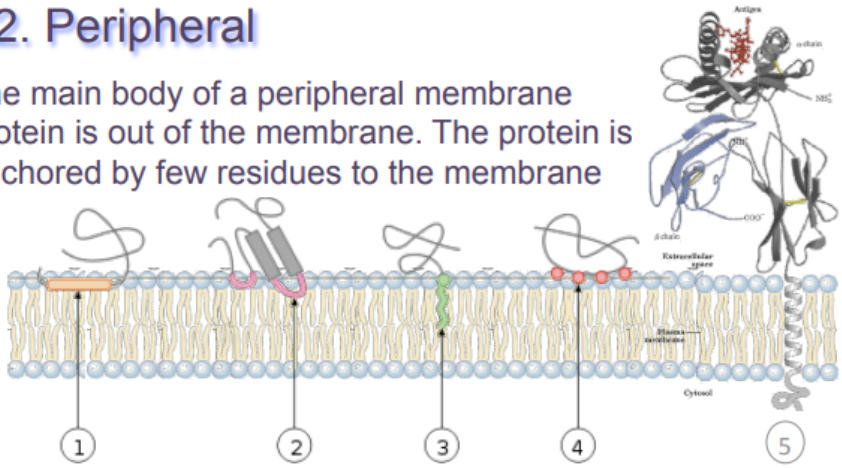
β-sheet transmembrane

Hydrophobic properties:

- α-helix transmembrane are all hydrophobic
- β-sheet transmembrane have hydrophilic and hydrophobic because they make a pore in the membrane. Meaning that one face is with the solvent.

2. Peripheral

The main body of a peripheral membrane protein is out of the membrane. The protein is anchored by few residues to the membrane



Example of different types of interaction between membrane proteins and the cell membrane: 1. interaction by an amphipathic helix parallel to the membrane plane 2. interaction by a hydrophobic loop 3. interaction by a covalently bound membrane lipid (lipidation, i.e. myristilation) 4. electrostatic or ioinic interaction with membrane lipids (e.g. through a calcium ion) 5. hydrophobic helix transmembrane.

Some proteins have two main bodies one at each side of the membrane, an in general its function is to transfer a biochemical signal from one side to the other

Comparative Modeling

Structure prediction based on homologs.

Basic Concepts of Homology Modeling

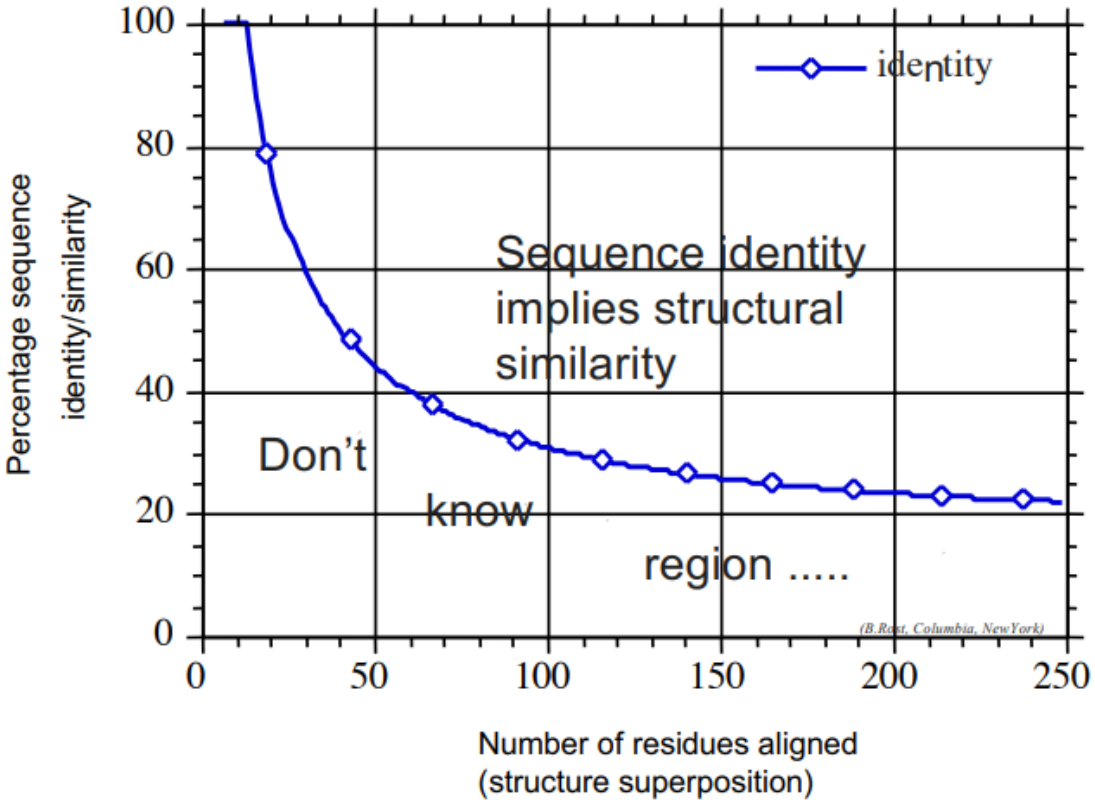
Also known as homology modelling → obtain the structure for a new (target) sequence from the known 3D-structures of related family members (templates)

Extrapolation of the structure for a new (target) sequence from the known 3D-structures of related family members (templates).

The number of different protein **folds** is limited. This tells us that, with the knowledge we have now a days, we can predict the fold of other proteins from which we do not know its structure. Imagine that the number of folds was infinite, we could not predict the folds of another protein because it may be a new fold that has never been studied.

Sequence similarity implies structural similarity?
We can superimpose structures to see if two folds are similar.
With this superimposition, we can calculate:

- How many residues can be aligned between proteins (X axis)
- How many of these residues are the same (Y axis)



Let's say that we have aligned 100 residues and we have obtained 80% of identical residues. Since it is above the curve of the plot, there is structural similarity. Meaning that you will be able to make a model.

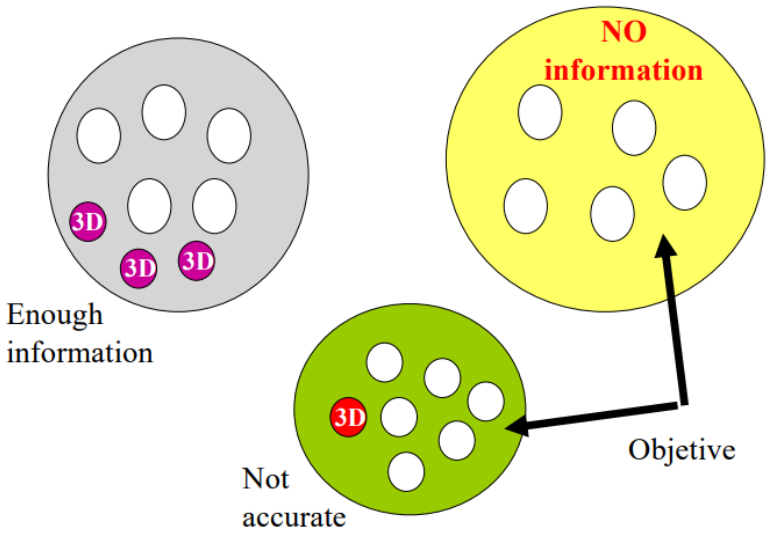
Fold is more conserved than sequence
Secondary structures are the most conserved parts, due to the hydrogen bonding so it contains the energy to fold.
Loops have higher variability in structure

Structural genomics

If we group by sequences by members of the same family, we have groups of sequences that are very similar between them (with many structures very well known), while there were other families with low or none known structures.

In this structural genomics initiative, it was thought that a way to improve the knowledge of the whole conformational space of protein structures was filling up the families without known structures.
Because in the families with known structures, we can obtain the rest of structures by homology, similarity…

We can not do this with families with a low number of known structures because there will be a lot of unknown regions and therefore it will be difficult to build the structure of the other proteins.
For a family without known structures, it is totally impossible.

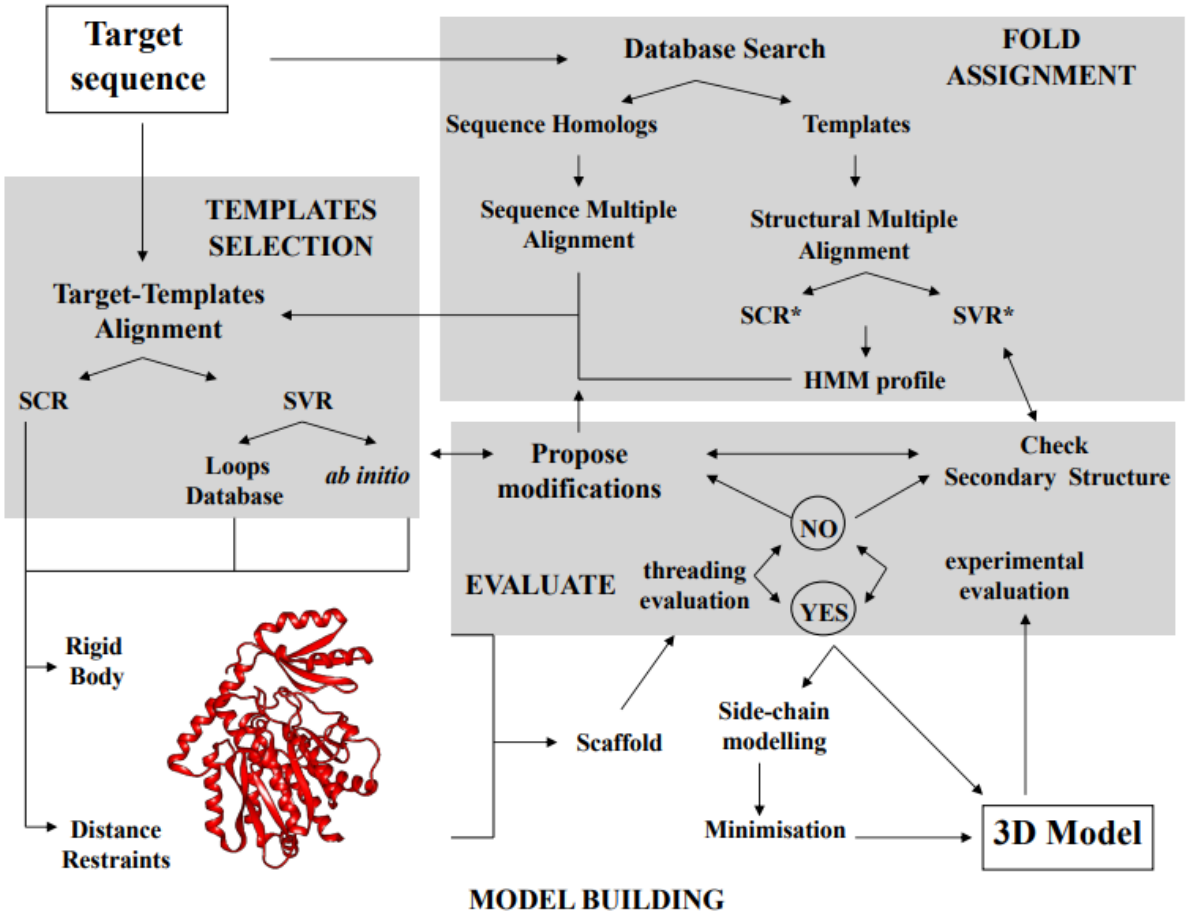


Schema of the method

The method to predict the structure of one protein based on homology modeling is based on the alignment of the target and template sequences. It has the following schema:

- Fold assignment
- Template selection
- Model building
- Evaluation
- Improvement

General schema



Sequence search with the target:

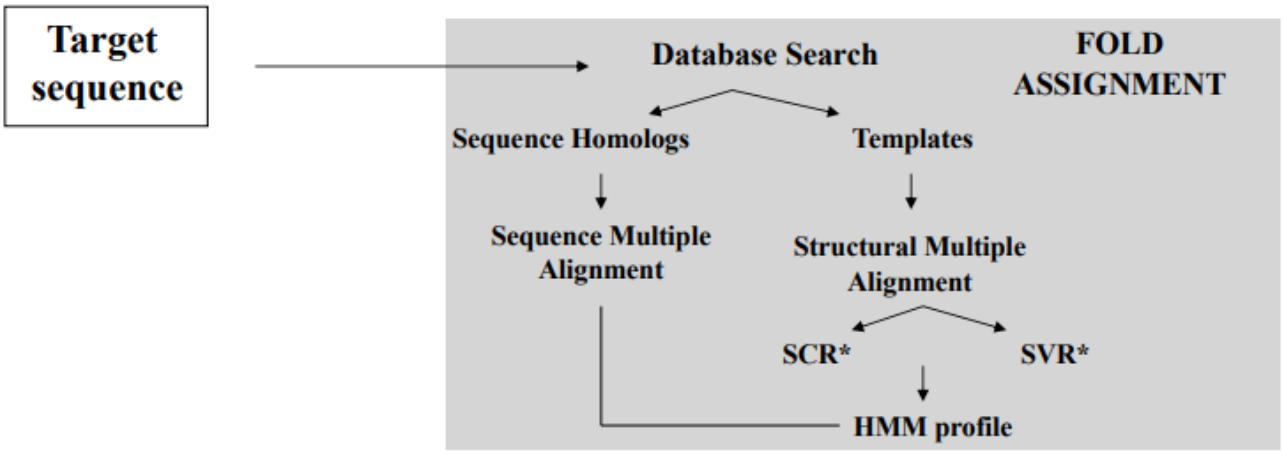
1. Compare the sequence of the target with a DB of sequences with known structure using some programs like BLAST, PSI-BLAST, HMM…
    a. We can obtain some potential Templates. Thus, we already know which is going to be the fold of our target protein.
    We can make a Multiple Structural Alignment by superimposing the structures.
    b. We can obtain sequence homologs and build a MSA instead of a structural alignment.

With both alignments, we can obtain a HMM profile. Thus, with the target we can search for all these sequences and with the HMM we can select the best Templates to make the modeling.

Then we construct the model and evaluate it.

- If it is correct, then we must refine it, by doing an optimization, analysis by molecular dynamics simulation
- If it is wrong, we will have to go back and make the rearrangements that we need. By changing the templates, alignments or profiles until we get a reniment of the structure and obtaining the model.

1. Fold assignment



Sequence search with the target:

1. Compare the sequence of the target with a set of sequences with known structure using some programs like BLAST, PSI-BLAST, HMM…
    c. We can obtain some potential Templates. Thus, we already know which is going to be the fold of our target protein.
    We can make a Multiple Structural Alignment by superimposing the structures.
    d. We can obtain sequence homologs and build a MSA.

2. Then we make a ranking according to some scores. These scores are based on how good the alignment is. We just look how many residues are similar in the alignment.

3. This score gives a P-values or E-values (high score implies low Pvalue). P-value is the probability of obtaining the same alignment by chance. Lower the value better the alignment

4. Scores are calculated using a residue-substitution matrix:
    a. PAM: based on the alignment of sequences of homologs (evo distance).
    PAM1 → 1 point accepted mutation per 100 residues (good for large distance)
    b. BLOSUM: based on the alignment of blocs of similar sequences

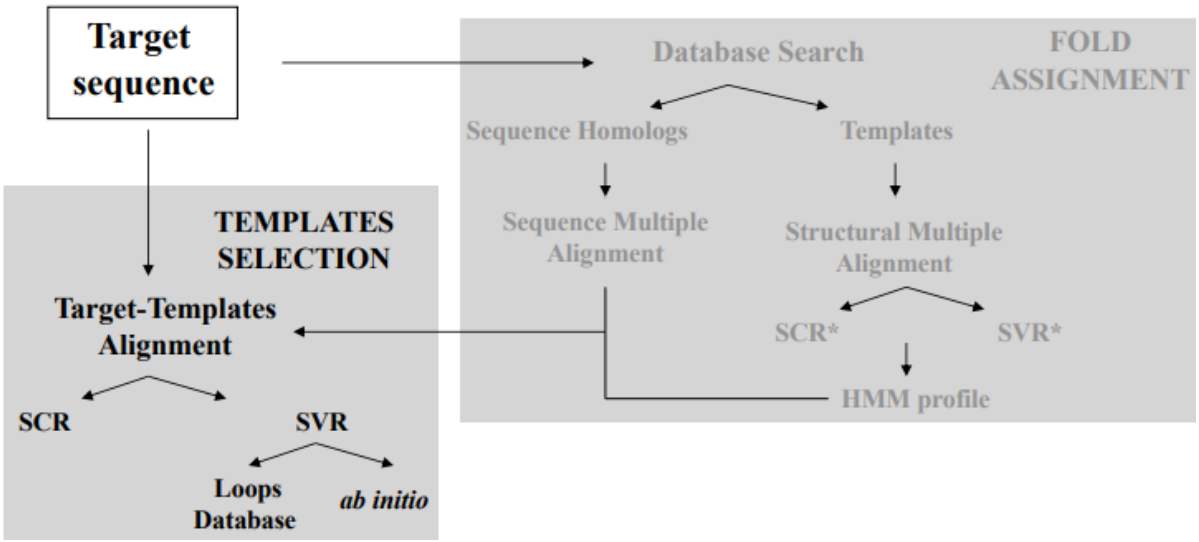
5. One sequence can have more than one domain, therefore we can obtain the best scores for partial parts of the target.

Methods (see practice)

- BLAST algorithm, matches words from a pre-calculated and indexed set and joints them into sentences (forming the sequence)
- FastA: Smith & Waterman algorithm
- Scanning PFAM: algorithm of Hidden Markov Models

So, with this step, we are obtaining a set of proteins which can be the templates from which we can do our modeling. We move now to the next step in which we select the best templates.

2. Template selection



Now we must select the best templates to make an alignment between our target sequence and the best templates.

Selecting the best target-alignment template:

1. The template(s) should be the closest homolog(s) to the target. Because along evolution it is going to be the one with less changes with respect to the target. So the structure will be much similar.

2. We must select a small number of templates (one template) to avoid stress on model building. So, we will get the templates that are the closest to our target.
   Once we have selected these templates, if we do a MSA with many homologs, we are going to see that there are regions that are conserved (SCR) and others that are variable (SVR) meaning that they have not been aligned with our target or even between the different templates.
   - Structural Conserved Regions (SCR)
   - Structure Variable Regions (SVR), which normally correspond to loops.

   How many close templates should we use? The best option is to choose a single template. Because if you use many, just because some structures are different due to the different resolution, you are giving information that can be misleading.

3. Multi-domain proteins require the use of at least one template with the largest coverage of sequence (containing the largest number of domains).
   So maybe one homolog is the closest to one domain and another to the other domain.

4. Detection of structurally conserved regions (SCR) and variable regions (VR).
   For SCR we can use this model but for SVR we must use other strategies.
   So, for SVR we do not have any template so we need to use other templates to fill the gaps.

5. Methods
    a. ClustalW
    b. T-coffee
    c. HMMER
        i. Alignment with a known family profile (PFAM)
        ii. Alignment with a profile built with the structure of homologs

Let's see what happens when we have more than one domain.

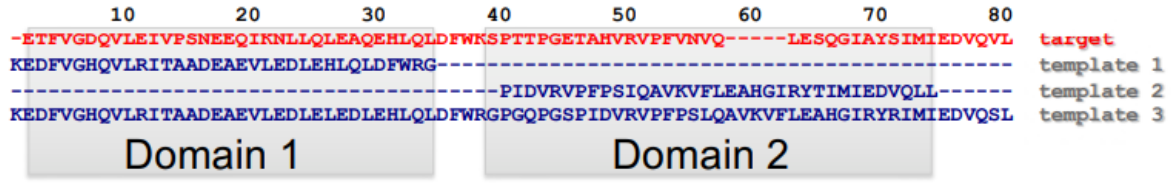
Maybe one template is very good for one domain and very bad for the other domain and another protein that is the opposite.

The problem is that maybe there is no link between both templates. Thus, we do not know how to place both domains (we do not know how they are connected).

Template selection, when you have more than one domain, and each target only has 1 of the domain, we need a third template that has both/all domains. Otherwise, we must perform docking, if not we cannot continue.

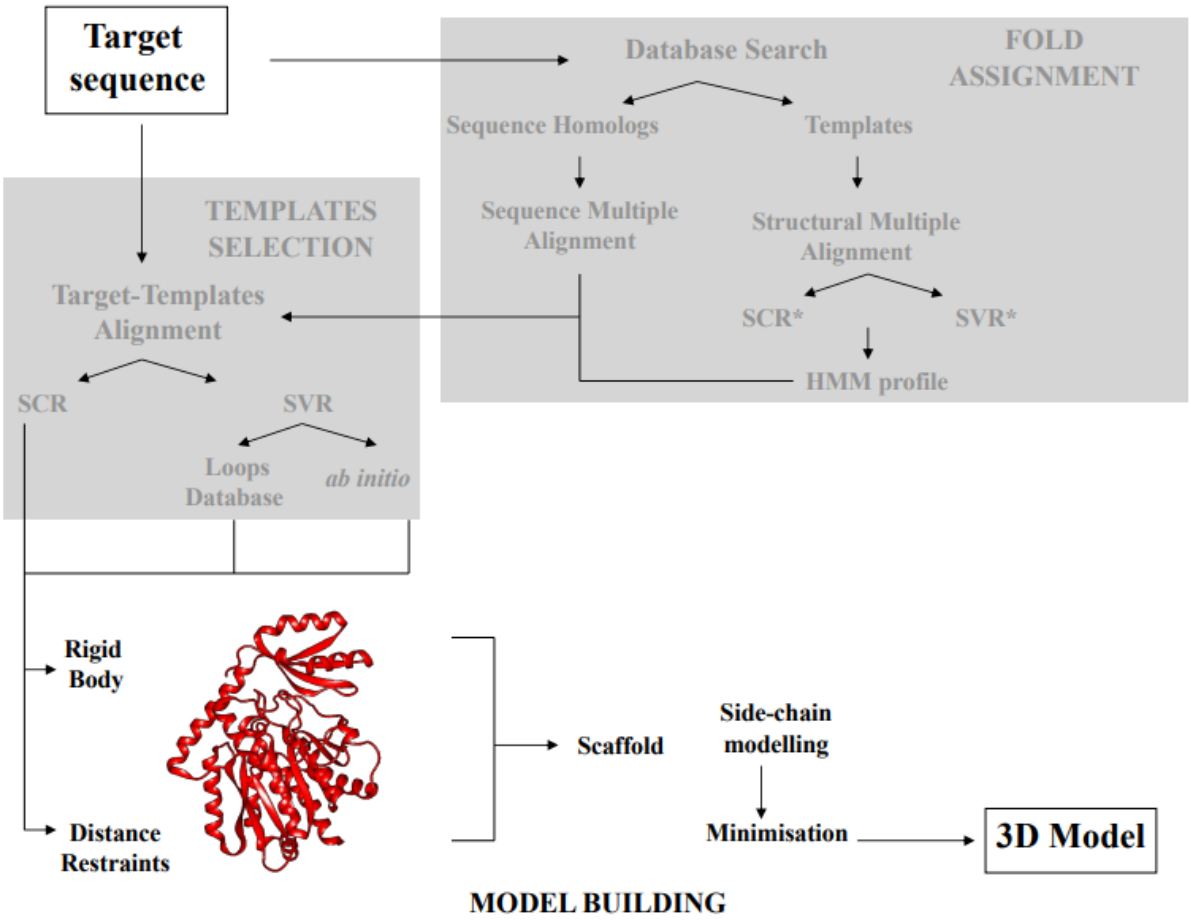
Since the third template will give us the information between the connection of domain 1 and domain 2. Sometimes it is a problem for those templates that are far away from the other templates and our target, so increasing the errors in our modeling and therefore we come up with a model that is not perfect.

Therefore, we need a third template that helps us to link both domains, even if this template is not very good.



This third template is normally very far away from the target protein and thus it can lead to errors.

3. Model building



MODEL BUILDING

Now we build the model and we have 2 approaches:

1. **Rigid Body Assembly** → We are separating the SCR and SVR.

    a. Core framework (SCR).
    We have an alignment between the target and several templates (minimum number of closest templates).

    - If we align these templates, we will observe the core of the structure (normally the regular secondary structures) that matches all templates.

      The closest the templates are between them, the best match we will get and if we take the averaging core template backbone atoms and we substitute it for the residues of our target.

      Then we have at least the structure (core) of our target.

    - There are some regions that are more variable and thus they are not superimposed by the templates. These regions remain unknown

    - Leave non-conserved regions (loops) for later…

b. Loop modeling (VR).
We need to fill the variable regions and to do this, we have 2 methods:

- "Ab initio" rebuilding of loops (Monte Carlo, molecular dynamics…) can be very difficult if the loop is very long. Often is better "Spare part algorithm"

- Use the "spare part" algorithm to find compatible fragments in a Loop DB. So, we try to find the closest sequences to our target and then pick up these loops and finish up the construction of the model.

  To fit the loop, we need:
  - Have a region that is really similar to our target loop
  - Have a very similar region in the N and C terminal of the loop

  We calculate the matrix of distances between the C-alpha atom and then we search for loops in the DB that have similar distance and similar sequences. Then, when we found one we forced the superimposition using the distance based matrix.
  We need to take into account that the angles and distances will not be the correct ones so we need to optimize it.

c. Energy minimization
To do this optimization we have to use programs like molecular dynamics simulations…
We use the potential energy with the bonding terms that find which are going to be the distance between atoms, the angles between 3 atoms, the dihedrals, electrostatic interactions between different charges, VdW contacts…

This is how you describe the energy of the system, but for optimization you need methods like: Newton Rapson; Steepest Descent; Conjugate Gradient.

Steepest descent → Go to the direction of the derivative of the energy to reach the minimum. Important to not take big steps or we won't ever end.
Conjugate gradient takes the information of previous steps so that this does not happen.

2. **Spatial/distance restraints.** We do not separate SCR and SVR, we just require the introduction of some spatial restraints.
    a. Probability Density Functions (PDF).
    We need to add local constraints, which are distributions of certain features. For example, a feature can be the Ramachandran plot, which shows the distribution of phi and psi angles. Certain positions of the angles are very populated.

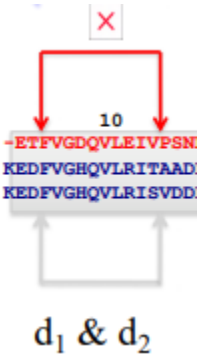
    Another example could be the distribution of C-alpha with C-alpha distances (2 gaussians), which brings us the probability of certain distances with respect to others. This can be transformed into energies using Boltzman equation
    $p = \exp(-E/kT) / Z$
    So, there is a relationship between energies and probabilities.

    We have the probabilities of the distance, so we can obtain the energy of the interaction between 2 C-alphas.

        i. This is what modeller does.
        According to the distributions seen in the PDB, modeller introduces these local restraints ($E_{pdf}(x) = -RT\log(p(x))$) for the interactions between residues based on probabilities . Most probable distances lower energy

    b. Distance restraints
    There are other spatial restraints that can be derived from the templates because we have an alignment between the sequence of the target and templates and we know the structures for the templates.

    Let's take into consideration the distance between the following F and P.

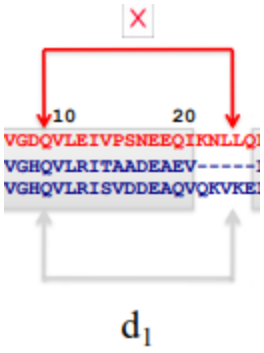
    

    We do not know which is going to be the distance because we do not have the structure of the target. But we have the structure of the templates, so we can assume that the distance will be:
    - The same as template 1 → d1
    - The same as template 2 → d2
    - The average

    So, we are adding a spatial restraint, which is the distance between two atoms. We can also do this for residues that are really far away in the sequence.

    If we have 10 templates, we will have 10 distances (if there is no full consensus between them) and we will be adding noise to the construction of the model.

We can also put restraints for the SVR.
In this case, we only have one distance since the other template has a gap.



d_1

So, if we missed template 2, we would miss this information and therefore we would have problems. Therefore it's important that we have enough templates so that we fill up all the gaps.

If the target sequence is the one that has the gap, we do not really care, because we won't have to put a restraint between the atoms because they do not exist.

c. Simulated Annealing
At the end we can obtain the structure from the alignment, just using optimization, taking into account all energy terms, and add 2 new terms:
- Distance restraints extracted from the templates (parabolic term) forcing them to have the same distance that we have seen in the templates.
- Local terms, which come from the tendency of local features to be maintained.

There is one problem here. If we try to optimize the energy using steepest descent, for example, it is going to crash.

Solution:

i. It is just not optimization, it is also molecular dynamics (the structure needs to move). So we use simulated annealing
We know that the total energy, which is constant, is the potential energy plus the kinetic energy.

Thus, if the kinetic energy is 0, we are moving on the potential surface. Kinetic energy is proportional to temperature.

ii. Simulated annealing, trying to find the point, we run this a lot of times trying to find the optimal folding.

iii. This is what **Modeller** does, increases the temperature, decreases, optimized, several times until ideally we find the structure for the model.

Per optimitzar el model, fem el simulated annealing, un procés iteratiu que permet a la proteïna plegar-se de mica en mica. Es fa la optimització de l'estructura al mateix temps que es construeix el model.
El procés d'optimització pretén arribar al punt de mínima energia del sistema (la vall més marcada del gràfic). Cal per això, passar primer per un pic d'energia que se situa per sobre de l'energia total, i l'única manera de fer-ho és amb dinàmica molecular : augmentar la temperatura del sistema provocant que augmenti la velocitat dels àtoms i per tant la seva energia cinètica, augmentant l'energia total del sistema. Els canvis de temperatura serveixen per anar provant totes les conformacions possibles fins arribar a la correcta.

d. Loop modeling
What happens in the variable regions (have no templates…)? In the case of Rigid Body Assembly we had to look for a DB of loops.

Can we do something similar in modeller?
Yes, instead of matching by superimposition, we use an alignment.

So, we align our main template and a second template that only contains the loop. This will tell the system what are the distance restraints for the construction of the loop.

It is important to notice that in this alignment we need to align the flanking regions of the loop (N-terminal and C-terminal).
The reason is that these flanking regions will tell the system how the loop is going to be placed.

Also, for the modeling of the loops, Modeller can also do Ab initio.

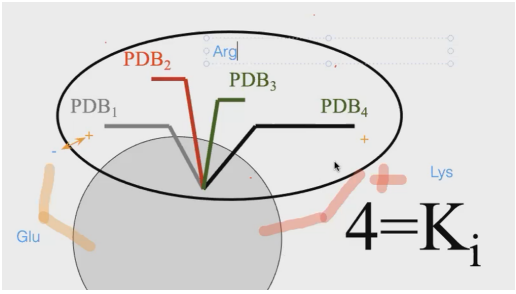
3. Side-chain modeling

Once we have the backbone for the structure, we have to model the side-chains.

a. There are models that add the side chains randomly

b. Back-bone dependent rotamer libraries
   i. Placing the side chains using the probabilities for the different rotameters.
   ii. The method uses a DB of rotamers (all the different conformations that we have in the PDB for the locations of the side chains).
   The probability of placing a rotamer is 1/all_rotamers for each side chain (CM).
   Let's say that we have an arginine, which has a very long chain with + charge at the end. We have 4 potential conformations to place this arginine → Probability is 0.25.

És un mètode que s'utilitza per al modelatge de proteïnes i es basa en la posició de les cadenes laterals. Es tracta de determinar els millors rotàmers (conformacions o angles en què es poden orientar les cadenes laterals) de cada residu de la nostra estructura en funció de la informació trobada en bases de dades (biblioteques de rotàmers).
S'agafa un residu concret en quatre posicions diferents, i per tant la P de tenir una posició concreta és 0.25.



What happens if in the structure I have a glutamate (with - charge) that makes an interaction.
On the other side there is a lysine (+ charge) that makes a repulsion.

So the most probable conformation is the one in the PDB1.

So, we need to add this information, which can not be found in the DB, it comes when you search the position of the side chain in the structure that you have modeled.

So, first I need to describe which will be the energy for each one of the different rotamers in a particular position.
So, the program is calculating the energy of the whole system once you have selected for a X residue one position. Then, this energy is calculated as the sum of all the interactions multiplied by CM.

4. Evaluation

We have placed the backbone, we have placed the side chains and now we need to evaluate the model created.

There are different kind of errors that we need to take into account:

- **Errors in side-chain packing:** If we apply the multicopy approach or a molecular dynamic simulation, we won't have this problem

- **Shifts of correctly aligned residues.**
  Let's imagine that we have the following sequences and we perform a MSA.

```
HHHHHHHH HHH .HHC
GARFIELD THE .CAT
GARFIELD THE CCAT
```

As we can see, we have introduced a gap in our sequence and thus, "CAT" aligns correctly. All MSA programs will give this solution, which is structurally wrong.

If we look at the secondary structure (H = Helix, C = Coil). We are introducing a gap in our helix and thus we are breaking H bonds.
Not only this, but the last 2 H will not make a helix… So, we have lost 3 potential H bonds.

So, we should obtain the following solution:

```
HHHHHHHH HHH HHC.
GARFIELD THE CAT.
GARFIELD THE CCAT
```

Even though we are missing some alignment, the secondary structure does not break. This is the worst solution in the sequence alignment, but it is the best solution for the structure.

The alignment of the residues was correct, but it was not good for the structure.

- **Regions without template:** We need to fix the gaps by introducing new templates that will tell us how the loops should look like.

- **Errors due to misalignments**

```
GARFIELD THE CAT ...
GARFIELD THE FAT CAT
```

Sometimes it is better to have a misalignment and the gaps at the end.

```
GARFIELD THE ... CAT
GARFIELD THE FAT CAT
```

But it is even better to align correctly and add the gaps in the middle.

HMM solves this by looking at the sequences of the members of the family.
T-Coffee solves this by looking at multiple sequences in a MSA

But if we only have 2 sequences, the programs are going to produce this misalignment. It is very difficult to solve.

- **Errors produced by incorrect templates:** This is the worst error and it is due to the wrong selection of the closest template. In this error we need to start again

Once we have the model, we need to test it.

1. Compare the RMSD between the model and the real structure. Just make a superposition of both structures.
2. Check that secondary structures are correctly aligned
3. Calculate the percentage of residues that are closer than a threshold after superposing the model and the real structure
4. Calculate the percentage of identical residues aligned when superposing the real structure and the model.
5. Check the energy of threading to compare the real structure and the model (see next chapter). Compare if the energy of the model is the same as the energy of the real structure.

5. Improvement

The problem is that you do not have the real structure, of course. So, you can not use this information to validate the model. So, this is another approach:

1. Compare the model and all the templates
2. Check that secondary structures are not broken
3. Check if the prediction of secondary structure agrees with the secondary structure of the model
4. Check if the loops of the target are similar to some loops in the database of loops and they agree in sequence and anchoring geometry
5. Check the capping of helices
6. Check pseudo-energies of threading and compare the model with the templates (will see later).

If there is something that does not match, we need to improve the model:

1. Decide the changes in the alignment according to the secondary structure prediction or the structure of the templates and recalculate the model (modify the alignment)
2. Change the main template and recalculate the model
3. Include new templates for the reconstruction of the loops
4. Calculate the main motion of normal modes from the templates of the homologous family and optimize by molecular dynamics under motion restrictions the conformation
5. Recalculate the pseudo energy profile of the new model and compare with the original model to test the improvement

Fold Prediction

This is a different case in which we try to predict the structure of a protein when we don't have homologs or similar templates.

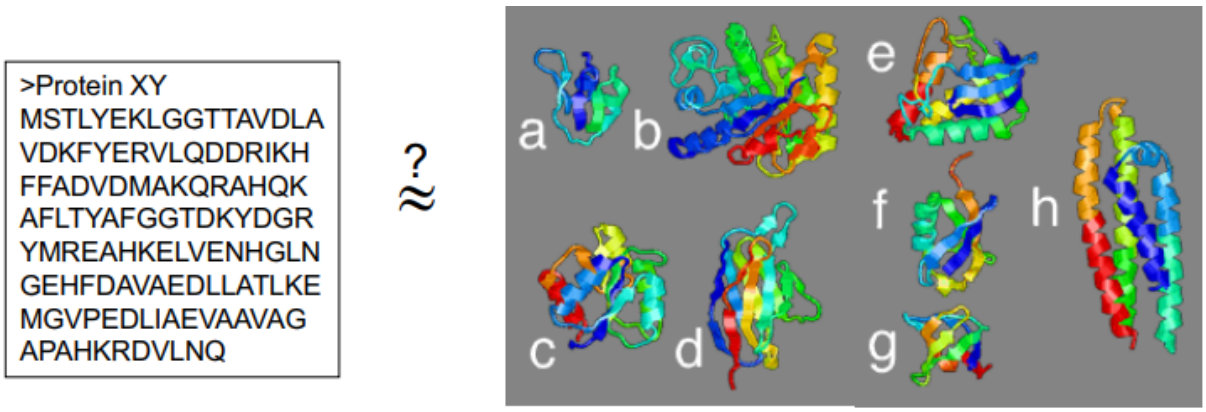
There are 3 main ideas when we talk about fold prediction:

- **Fold recognition (threading):** Recognizing what type of fold a protein has.
- **Ab Initio prediction:** Predict the fold of the protein even if this structure does not exist in the DB (totally new structure).
- **Protein Folding:** How is the protein getting the conformation in 3D and usually this is done by MD simulations with explicit solvent. This is an unsolved problem.

1. Threading

Find the optimal structure for a new (target) sequence in the set of known 3D-structures (templates) by threading the target sequence.
Try to infer which one is the template that can be assign to the target sequence

It's called threading because it places the sequence in all the different folds and then selects the best fold. So, we have a target sequence and we try to infer which one of the different folds is the best one.



To decide which one of the folds is the best one, we need a scoring function.
This scoring function is **Statistical Potentials.**

Not an energy function based on the classical energies that we have been using → We will get a good score but a really wrong fold!

1.1. Knowledge-base potentials

1.1.1. Distance dependent potentials

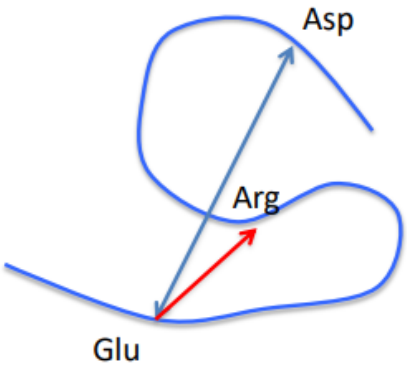
What is an statistical potential?
According to Boltzmann law:

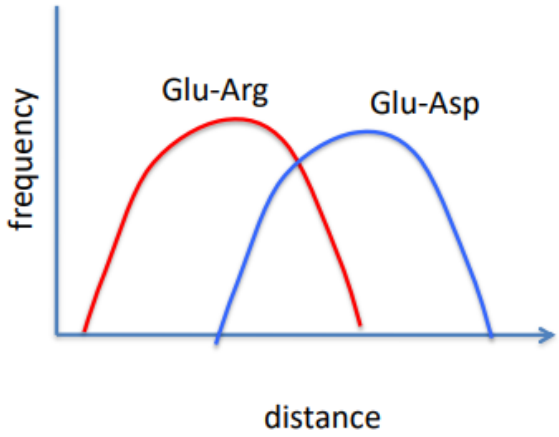
$$P(x) = \frac{1}{Z} e^{-E(x)/k_B T}$$

The probability of one microstate depends on the energy of that microstate.
So, this function shows a relationship between probability and energy.

Meaning that i can extract energies from the probabilities
So, we are going to create a potential based on the probabilities (probability that certain pairs of residues interact).

Let's think of the following 3 residues:





Asp has a negative side chain
Arg has a positive side chain
Glu has a negative side chain

What is the expected distance between these residues?
Asp and Glu, ideally will be separated because there is a repulsion between their side chains.
Glu and Arg should be close for the same reason,

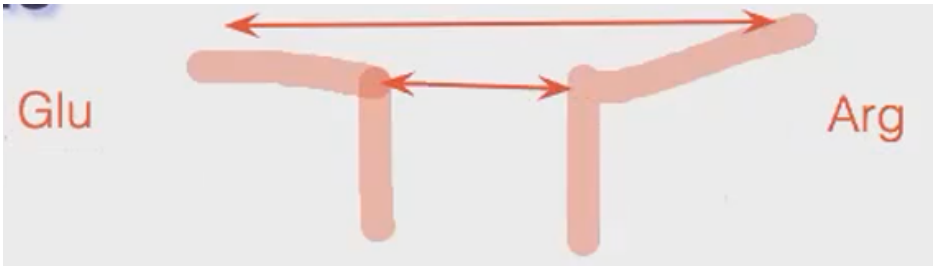
We can see this in the plot → On average, Glu-Arg are closer than Glu-Asp

When we are calculating distances, what distance do we calculate?

- Distance between C-alphas?
- Distance between side chains?
- Distance between C-betas?

Distances are calculated between atoms: We have to select what atoms are we going to use.

- The best choice is C-beta because it indicates the direction of the side-chain



The C-alpha (middle) does not reflect the direction of the side chain. Meaning that all possible conformations of the side chains are equally good, which is FALSE.
We know that they have opposite charges, so actually the chains should be looking at the same place (attraction)

The C-beta does represent it, so if the distance is really big, it means that they are looking in opposite directions.

Note that Gly is the only atom that does not have a C-beta, only C-alpha.
In this case we should use C-alpha, but we won't know the direction of the side chain.

When we select all the proteins from the PDB, we must take into account that some proteins have been crystalized very often.
The DB of structures to extract distances has to avoid redundant structures (between homologs and members of the same family/superfamily).

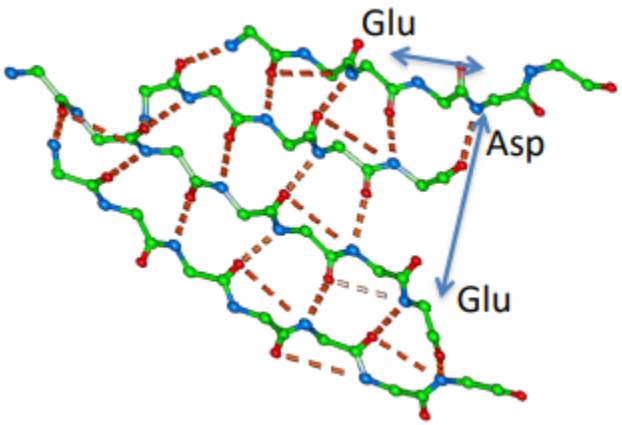
- Thus, we are calculating all the time the distances between residues that correspond to this family of proteins. This is a problem because it creates a bias.
  When we try to infer what is the energy, stability, scoring function… of a protein that has nothing to do with these popular families, we will get a different result from what we expect.
- So, we get rid of the bias by reducing the dataset to those that are non-homologs.
  Thus, we use a set with sequences with less than 40% of sequence similarities.

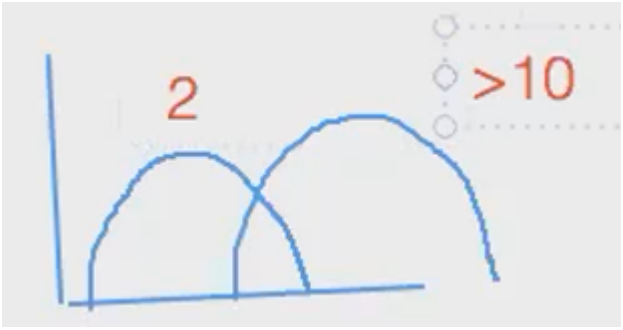
When we calculate these frequencies of distances. Let's use again Glu, Asp and Glu:

We have an Asp (100) near a Glu (101) and another Glu (200) that is very far in the sequence.

When we calculate frequencies, we are expecting that these Asp is far away from the Glu (200).

But when calculating the distance between the Glu (101) and Asp, they should be close since they are connected through the sequence.





Thus, when calculating the frequencies we must take into account if the residues are close in the sequence or not.

So, the frequency of a pair of residues at distance "r" is different if the residues are close or distant along the sequence.

- We split the calculation of frequencies depending on the sequence distance between residues. It is not the same to calculate the frequencies from residues that are close in the sequence and those that are separated in the sequence.

We also need to take into account the **reference state**.
It is not the same to calculate the frequency of residues around a residue that is in the interior of the protein or in the surface.
The density of residues around one residue is not a continuous model, it depends on the size and shape of the protein. So it will not be the same frequencies if we are calculating 1 residue around if we are in the interior of the protein rather than if we are in the surface of the proteins.

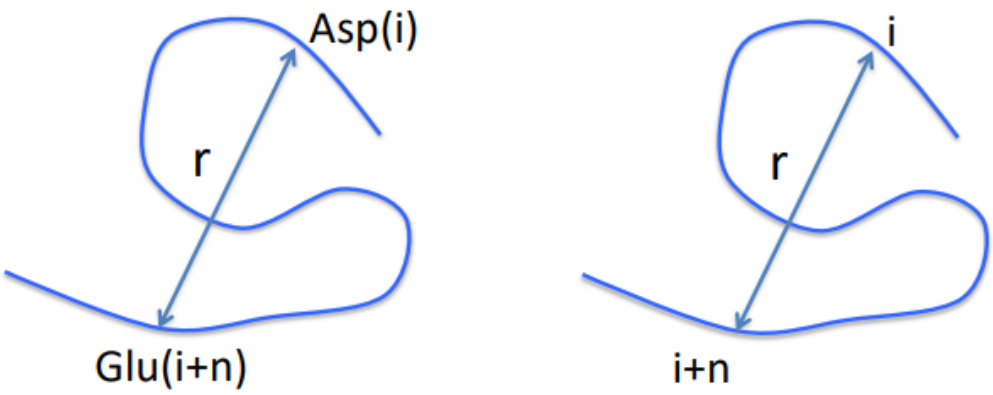
Thus, we need to take into account the density (residues that are around) in order to calculate the proper frequency.

- We need to normalize by the density ($4 \cdot \pi \cdot r^2 \cdot \varepsilon(r)$) and thus defining a reference state

  There is a very simple way to do it. How many residues we have around a particular residue and then we can calculate this as this reference. Then we calculate the frequency of a certain pair of residues with respect to the reference.

  It would be something like this:
  If we are going to calculate the frequency for the interaction between an Asp and a Glu, we are calculating how many aspartates and glutamates are at a given distance taking into account the number of residues between them in the sequence.
  Then we normalize this (we divide it) by the frequency at which you will have any pair of residues, not just Asp or Glu.

  Taking all this into account, the formula to calculate the statistical potential would look like this:



$$\Delta E(r/(Glu,Asp,C\beta,C\beta,n)) = -kT \ln\left(\frac{N(r/ED,n)}{N(r/n)}\right)$$

N(r/ED,n) → Frequency that we have seen Glu and Asp at a given distance, when they are separated by n residues in the sequence.

N(r/n) → Total number of any pair of residues that we have at this distance.

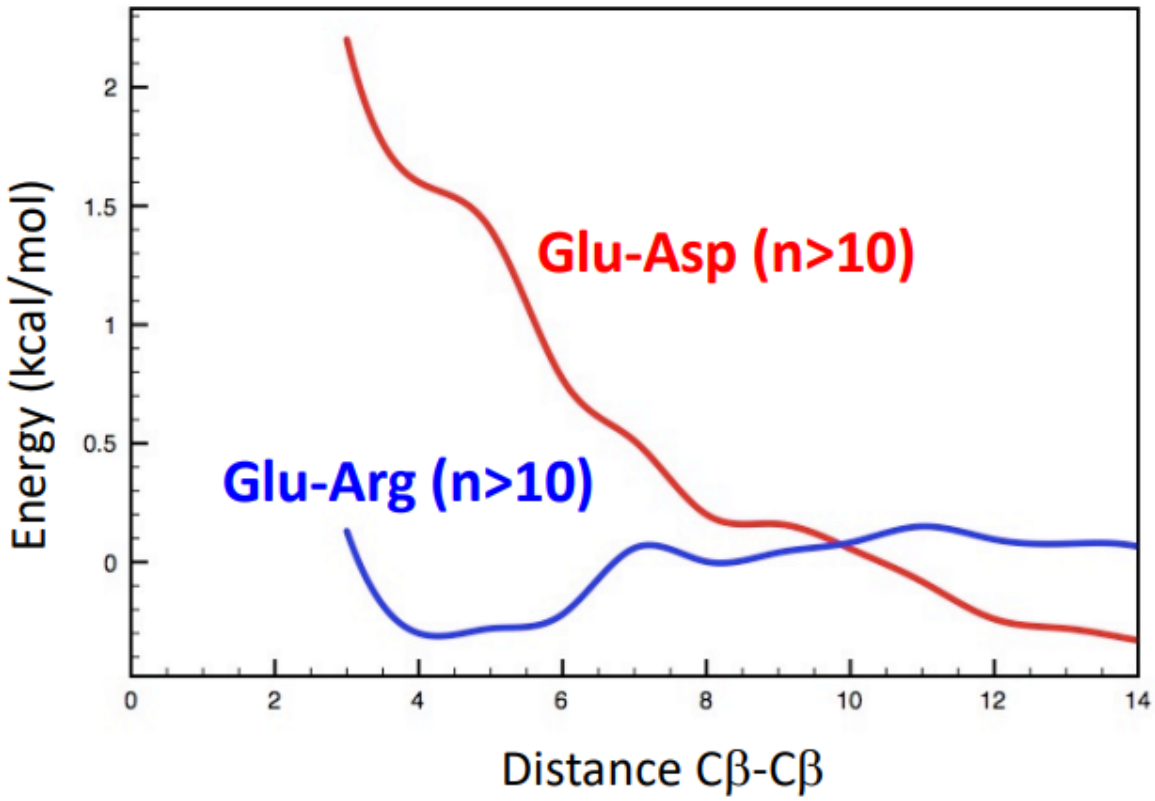
This is going to be the statistical potential for the interaction between the Asp and Glu at a given distance.

Let be a pair of residues Asp and Glu at distance n in a sequence. Let be N(r/ED, n) the number of pairs ED like this at distance r between their C-beta atoms, and N(r/n) the total of pairs of residues at distance n in sequence and r between their C-beta atoms.

Example of distance dependent knowledge-based potentials:

This can be calculated from a database from PDB, reducing by 40% the sequences.
We take the frequency of Glu-Asp and Glu-Arg at a larger distance of 10 residues.

We can see a minimum around 4 and 6 A for Glu-Arg, while Glu-Asp are better separated in space.



1.1.2. Solvation

As some residues are on the surface, meaning that the interaction of these residues is not only with other residues but also with the solvent.

We need to take into account how stable is this residue when it interacts with the solvent:

- A polar residue likes to be on the surface in contrast
- Hydrophobic residues don't like to be on the surface.

We know that there is an energy of solvation that is proportional to the atomic Accessible Surface Area of the atoms of the residue.

$$E_{sol}(i) = \sigma_{Asp} ASA(i)$$

The factor of proportion depends on the tendency of the residue to be solvated

- If the residue is polar, then the proportion will be negative so it is stable.
- If it is a hydrophobic residue it will be positive (positive energy is bad).

Solvation can also be calculated using the frequency of the residue to be exposed on the surface.

The important thing is that in both strategies, they have different magnitudes, so we need to have a way to balance both different type of energies

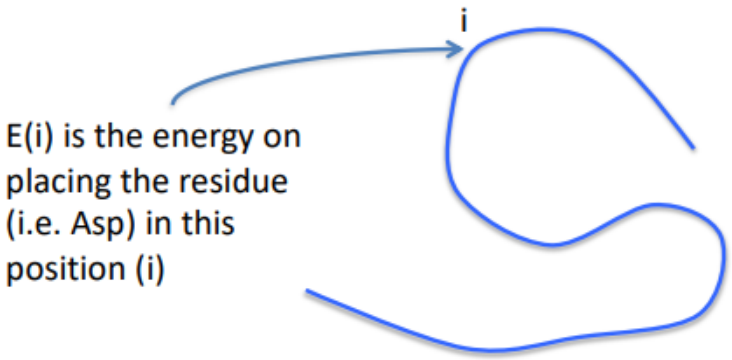
1.1.3. Z-scores and energy profiles

When we calculate what is the score of the energy when we place the sequence in the structure, we do the following:

- We take into account what is the energy for each one of the single residues.

Imagine residue in position 'i' → What is the interaction for this residue? It will be the sum for all the interactions of this residue with the rest of residues plus the interaction of this residue with the solvent.

So, once we have a set of energies for pairs of residues (force field) we can calculate the energy of each residue along the sequence in a specific conformation



So, the formula should look like this:

$$E_i = \sum_{j \neq i} E_{ij}\left(r, n = |j - i|\right)$$

$$E_{ij}\left(r, n = |j - i|\right) = \Delta E(r/(Glu(j), Asp(i), C\beta, C\beta, n))$$

$$E_{sol}(i) = \sigma_{Asp} ASA(i)$$

Ei → All the interactions with the other residues , which can be calculated using the statistical potentials (Eij)
Esol → All the interaction with the solvent, which is calculated with the ASA

Note that Eij and Esol have different proportions and therefore we must solve this by multiplying Esol by another factor, for example.

At the end, the total energy of a protein is obtained by the sum of the pair-energies and the energy from its surface (solvation)

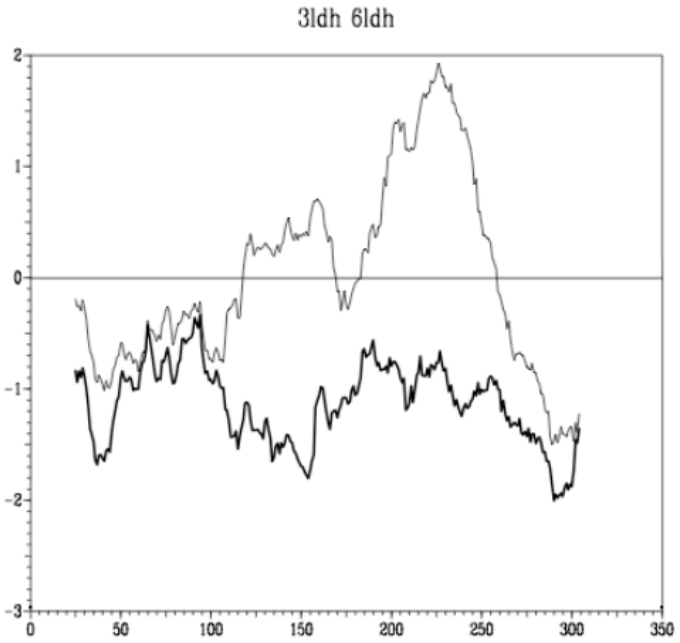
$$E = \sum_i E_i + \beta \sum_i E_{sol}(i)$$

Usually beta is ⅓ or ⅕

Then we have a kind of scoring of this energy → How probable are certain pairs of residues interacting.

In Statistical Potentials, energies means how frequent are the interactions between residues. So, a very good energy (low) means that the residues are paired with those that you have often seen in the PDB.

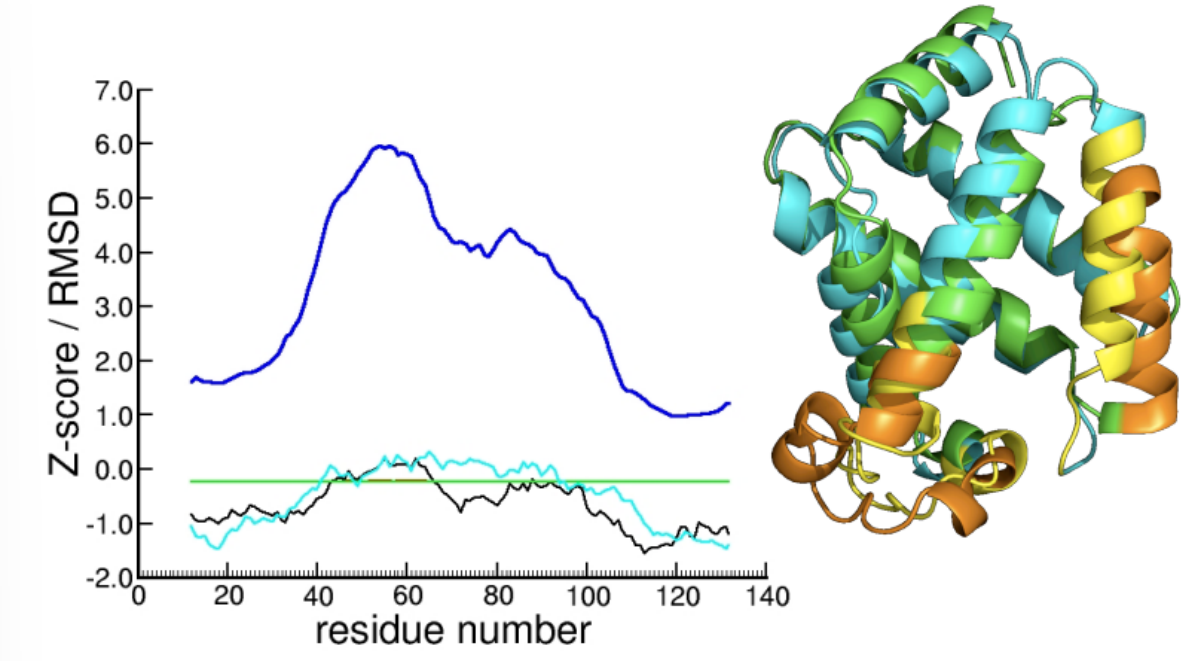
Example of profile energy from PROSA:
This is a profile of the energy in which we have the position of the residues (X axis) and then we can detect whether there is a peak, meaning that there is a wrong region.



Often the curve is smoothed by windowing the curve: the value on each point is defined by the average of a window of W residues and the window moves along the X axis.

From here we can know if our model is correct or not, if it is positive it is saying that the interaction between amino acids hasn't been seen in the PDB so the model is wrong, the negative one is correct.

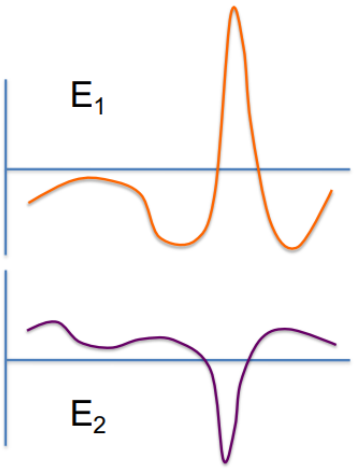
Energy profiles can be used to detect errors in modeling and therefore we can correct them later.



It also helps to determine whether one structure is the right one or not.
To make a decision whether one structure is right or wrong, we can use the whole energy:

- If I compare 2 possible models. Meaning that I place the sequence in 2 different structures. I would see something like this.

- We can see that energy for model 1 is higher than energy for the model 2. Thus, the 2nd model is better.



$E_1 > E_2$

Wrong solution

This is wrong! Because E2 is the best because of a single fragment. The rest of the structure is bad.
In the other case E1 is good for most of the structure and very bad for a fragment.

Question: Can we use the total energy to discriminate correct folds among wrong conformations (decoys)? NO

What happens if you have several cases and all of them have energies lower than 0 (they are all possible).

$$0 > E_1 > E_2 > E_3 \mathrm{L} > E_n$$

Many solutions is a wrong solution

We could take the one with the smallest energy, but this could not be right as we said before.

So, which one of these is the correct one?
This is solved by something that is called Z-score → Low Zscore will mean that it is the correct fold
Z-scores are used for normalizing values.

The best way to decide whether one energy or score is the best is by comparing them with respect to the average. If all of them are good, we must check which one is further away from the average getting the best energy.
The way to do this is to subtract the average and dividing it by the standard deviation.

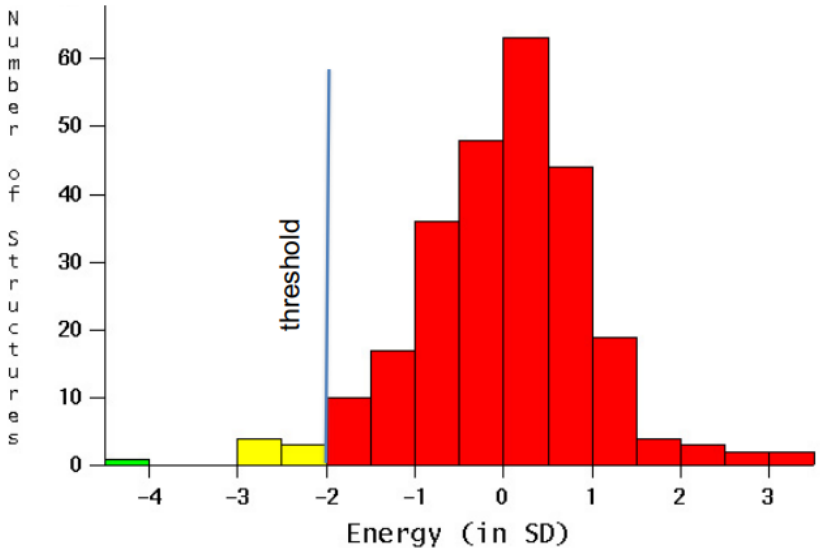
$$Zscore_j = \frac{E_j - \langle E \rangle}{\sigma}$$

Ej → Energy of our sequence with a particular fold
<E> → Average of the energy of our sequence in all the folds

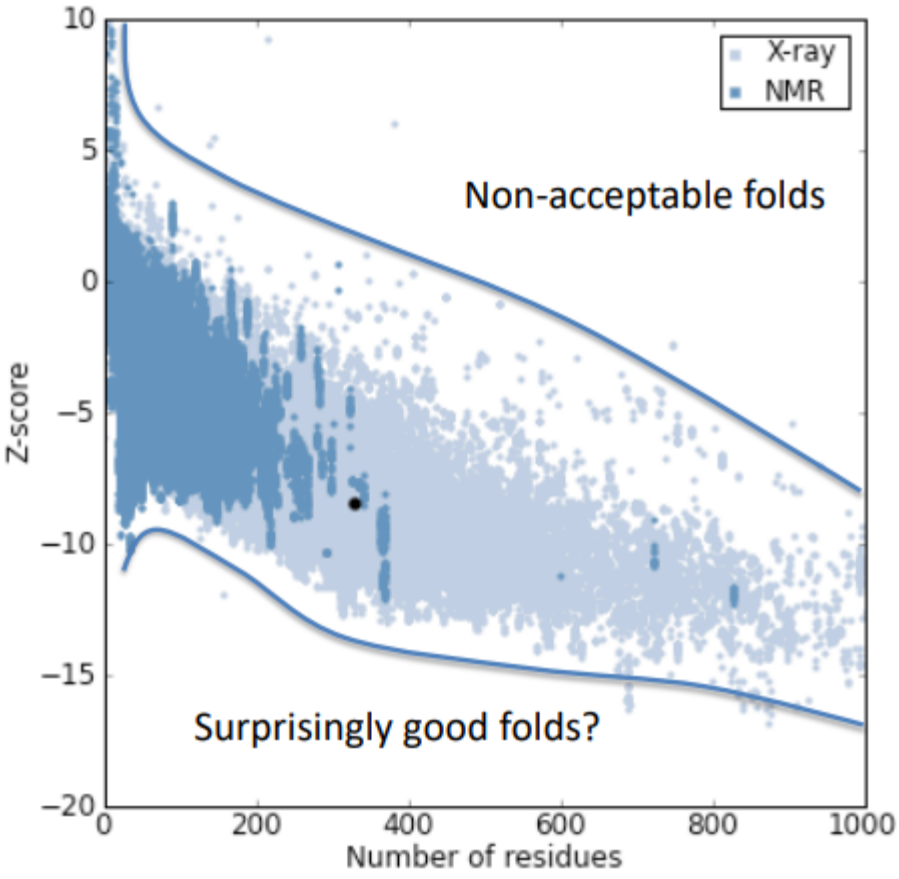
Threading Z-score is defined by comparing the energy on one fold (j) with the average of the real folds from the database (i.e. transforms the function "energy" into a Gaussian distribution centered at zero).

This is the same problem as the following: Consider the final marks in the class after the exam. We can calculate the 10 best alumni according to their marks. Are these the best alumni of SBI in the world? We have to weight their marks with the best students of the world, assuming the exam was the same. To do that, we use the set of marks of the total of SBI teachers in the world, and we assume they are the best set. Then, we compare our 10 alumni with them. If their marks are similar (close to the average of teachers), they are indeed the best.

This is a classical example of threading in which you select the proteins that are far away from the average with enough standard deviations.



Z Scores can also depend on the length of the protein. It is not the same a Z-score of -4 for a sequence of 200 residues and 1000 residues



Remote homologs (PSSM) Use blast or hidden markov profile

This is how threading works: you select one fold, test the sequence, place the sequence in the fold, check the energy and then select the best one.

There are other approaches for obtaining potential folds for one protein. For example, we can use:

1. Alignment between sequences using PSSMs (BLAST)
2. Alignment between one sequence and a Hidden Markov Model profile (hmmpfam, hmmscan).
3. Alignment between **two** Hidden Markov Model profiles (HHSearch, PRC). We compare the HMM of the target sequence with the HMM of the sequences with structure.

This is very good to infer remote homologs (they are very far away along evolution). Impossible to compare them using classical sequence alignment, because they are too different. So we must use those matrices to have a relationship between the sequences.

1. Function Association

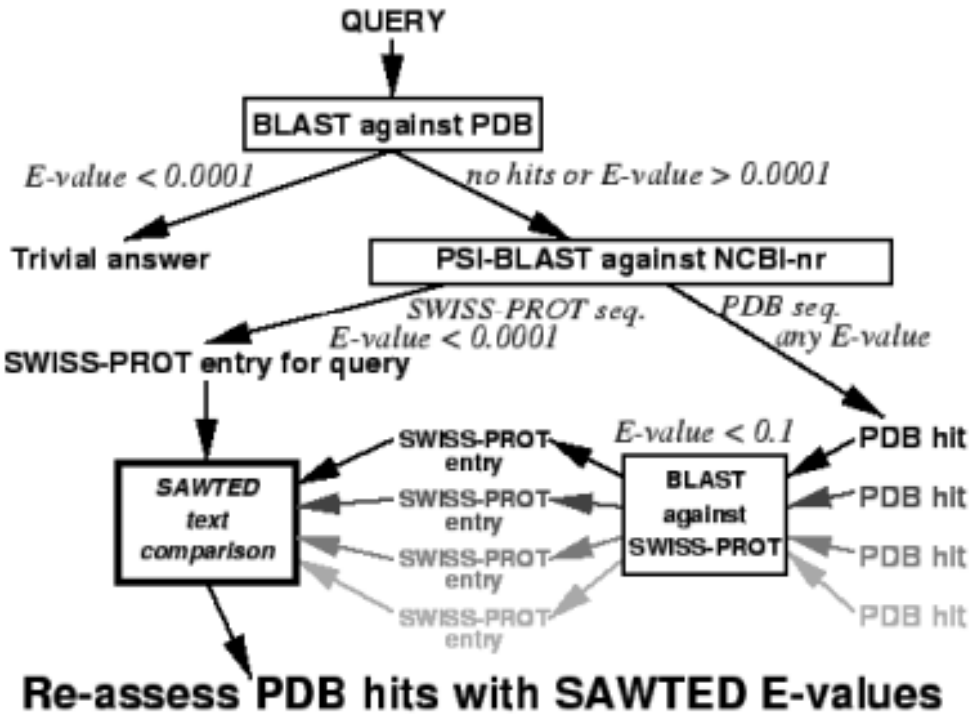
There is a particular case of a method called PHYRE that uses this kind of PSSM but it uses an additional element to distinguish between good and bad folds. Uses a lineage processing called SAWTED.

How SAWTED works → Imagine a protein from which you do not know the fold.
The possible folds are Kinase, Globin and Ig (very different folds).

If they have very similar bad E-values, we do not know which fold is the correct one.
If the E-value is correct then we know the answer.
Well, SAWTED checks if in the BLAST output the word “kinase” appears many times. If it does, then kinase will be the fold.

SAWTED stands for **S**tructure **A**ssignment **W**ith **T**ext **D**escription. It is a method to improve the coverage of the detection of remote homologues of known structure by sequence searches (e.g. PSI-BLAST) and fold recognition programs.

What does it do? It extracts the information of the key words that appear in the BLAST output.



We have a query and we run BLAST against the PDB:

- Good E-value → We know the solution
- Bad E-value

Run psiBLAST in SWISSPROT and select all the potential templates that correspond to my protein, calculate the PSSM.

On the other hand, with the PSSM we run in PDB and get the templates with the best E-values.

From all these different templates we select the ones that correspond with the same kind of functions as the ones detected by SAWTED.

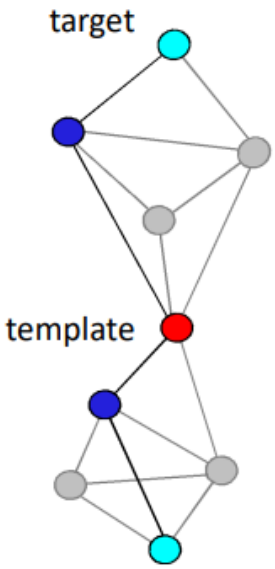
What would happen if this information does not exist. You run in UniProt and there is no information. You have to run in it on TREMBL, which contains millions of sequences but the information of the function is unknown.

Thus SAWTED is going to fail because it does not recognize an specific function.

We must use other approaches such as Modlink which uses information about protein protein interaction.

Modlink

Uses the knowledge of protein-protein interactions to select the best candidates (according to sequence-based alignments) among the homologs with known structure.



Nowadays, thanks to the high throughput methods, people are getting protein-protein interactions in a high throughput way. So, we can use this information:

Imagine we have a target (you do not know the structure, function…).

Let's consider a potential templates (which have a similar fold) as they will have similar functions so similar roles, they will interact with similar proteins. As they have similar folds, they will have similar surfaces, so they have similar partners so clearly they will have similar interactions and that is something we can check in the database. So see the interactors of the target and the template, and if they are common that means they interact with similar proteins and they are related.

In this case, the target and template are at a distance of 2 on the network.

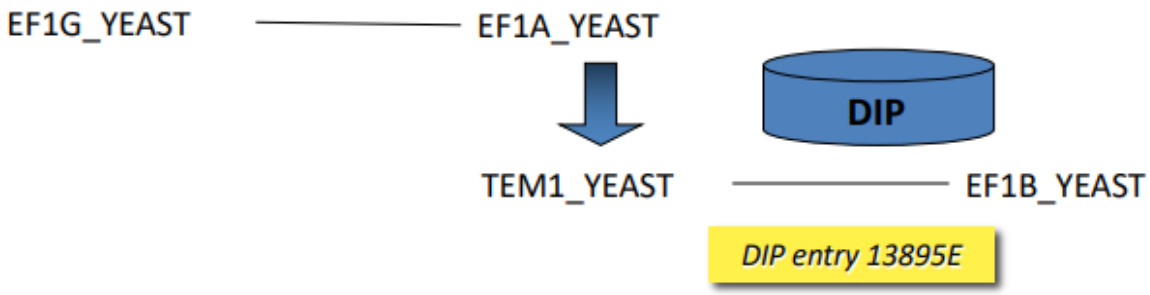
Example (blind test → we know the answer but we suppose we act as we do not).
We have a target that is similar to the following 3 proteins. As we can see, the E-values are bad so we can't trust them.

The right answer is the second one, but we should not know this.

PSI-BLAST search of the C-terminal domain of yeast Elongation Factor 1γ (**Ferredoxin like fold**)

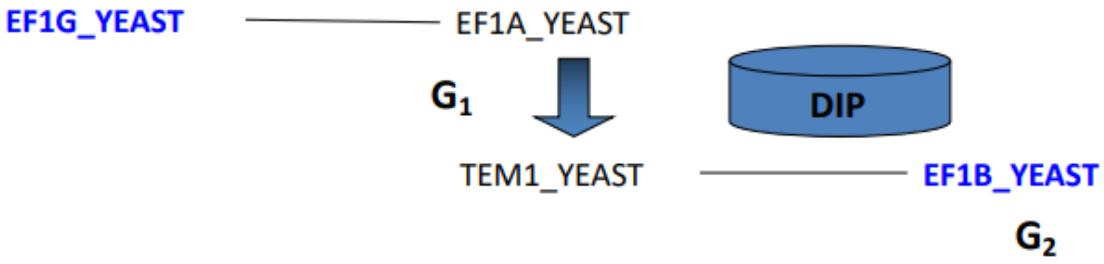
Hits in SwissProt	E-value	Shares Fold	Appears in G_2
SYEC_YEAST	0.027	no	?
EF1B_YEAST	0.036	yes	?
SC14_YEAST	0.83	no	?

Now I have to check which is the distance in the network.



As we can see, the correct template it is at distance 2.

Hits in SwissProt	E-value	Shares Fold	Appears in G_2
SYEC_YEAST	0.027	no	**no**
EF1B_YEAST	0.036	yes	yes
SC14_YEAST	0.83	no	**no**



Thus, we expect that this is the correct answer.

3. Secondary structure alignment

3.1. Secondary structure prediction (Machine learning)

Besides assigning folds by similarity of sequences or functional relationships, we can also try to find a relationship based on secondary structure.

At the beginning, the methods of secondary structure prediction were based on how often certain residues are found in alpha helices, beta strands…
The best results were obtained using neural networks → Deep learning

All machine learning methods are using an approach based on probabilities (bayes theorem).

The bayes Theorem

Let's consider 2 sets:

- M → Set of data obtained with a predictive model (Models predicted)
- D → Set of known data (PDB structures that we know)

$$P(D/M) = \frac{P(D \cap M)}{P(M)}$$

$$P(M/D) = \frac{P(D \cap M)}{P(D)}$$

According to Bayes Theorem, we define first some conditional probabilities:

- Probability of having something true in the D set when we know the model
- Probability of having a true model when we know the data

$$P(M/D) = P(D/M)\frac{P(M)}{P(D)}$$

From this, we extract the Bayes Theorem → The probability of having the model given the data is equal to… P(D) = 1, because it is always true

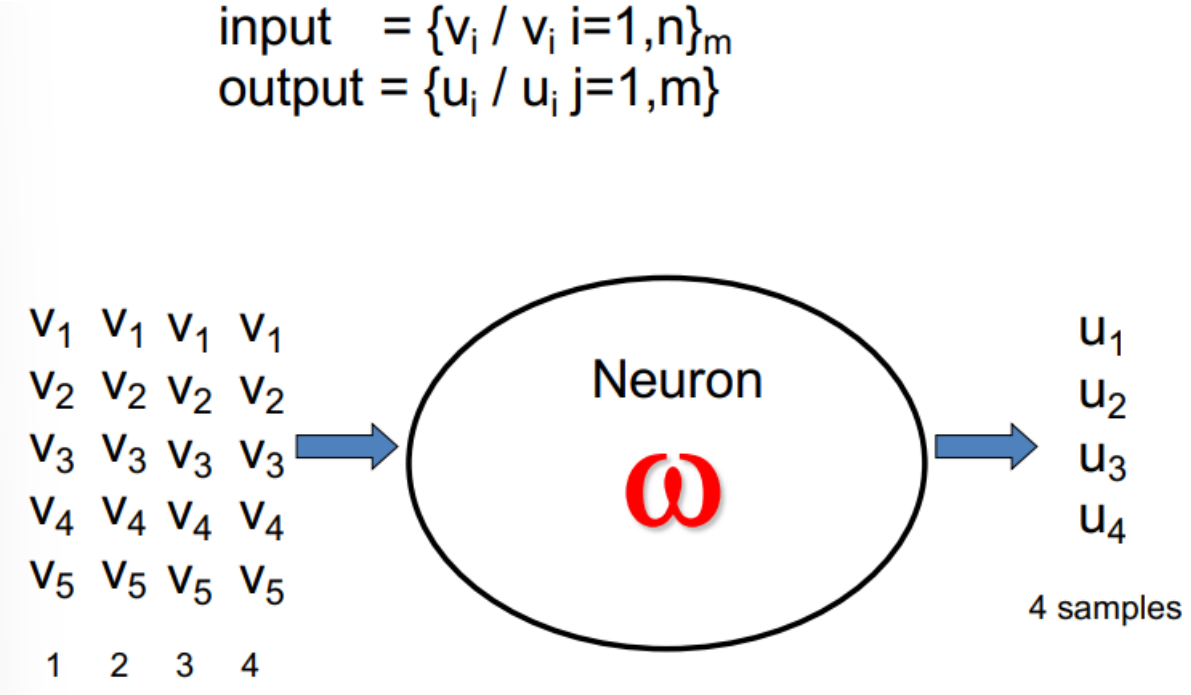
Instead of working with probabilities, we make the minus logarithm. It is easier to work with.

Now we obtain the maximum likelihood and we optimize it using steepest descent, for example.

Training set
Set of data without redundancies (i.e. a set of non-homologous sequences). This is used to optimize the parameters describing the model .

Test set
Set of data without any element used on the training set or similar to some element of the training set (i.e. a set of sequences nonhomologous between them and non-homologous to any of the elements of the training set). This set is used to test the approach and validate the statistical accuracy of the method.

Example of perceptron…



This is for the training set, several examples with some inputs, use it do something and transform it into the output and have an answer for each example
So we need to reduce the dimension of the input , so a function (the easiest one would be a linear where you sum the different values and from the vector obtain a single value; *es la primera formula*) but this will go from -infinity to +infinity and we want to control it for example that is 0 or 1 (like helix 0 or not helix 1).

So we want to transform it using another function (*es la segunda que va de 0 a 1)*. Then we need to optimize the parameters with the training set.

Parameters for the model: ω

$$x_j = \sum_{k=1}^{5} w_k V_k^j + w_0$$

$$y_j = f(x_j) = \frac{1}{1 + e^{-x_j}}$$

We need to optimize the parameters in order to get y_j as close as possible to u_j

So we will compute the probability of the data given the model in the most simple way using a gaussian distribution, where we have several values for each of the samples so it will be the product of all the gaussians assuming they are independent samples

Working hypothesis: The error between the expected output values (u) and the output obtained with this "neuron" approach follows a multiple gaussian distribution. Therefore, the probability to obtain the output data, given the parameters of the neuron (w and function f), is:

nd function f), is:

$$P(D \mid M) = P(u \mid \omega, f) = \prod_{j=1}^{m} \frac{1}{\sigma\sqrt{2\pi}} \times e^{\frac{-(u_j - y_j)^2}{2\sigma^2}}$$

$$\sigma = \sqrt{\frac{\sum_{j=1}^{m} (u_j - y_j)^2}{m-1}}$$

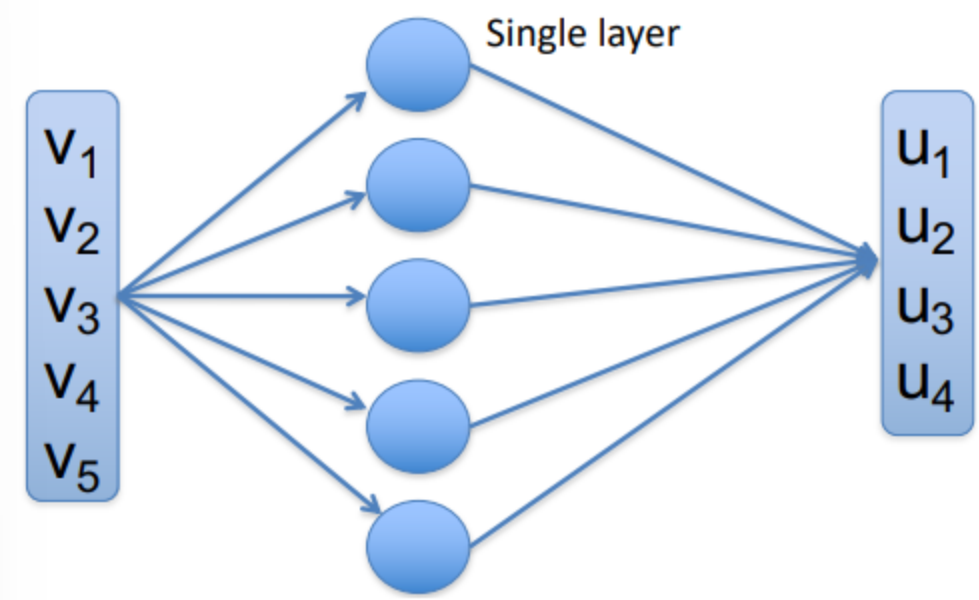
Where m is the number of samples

Maximum Likelihood solution: This implies we can solve the optimization by means of the maximum likelihood approach. It also can be further simplified by assuming a constant standard deviation.

$$\Phi(w) = -\log(P) = -\frac{\log(2\pi)}{2} - \log(\sigma) + \sum_j (u_j - y_j)^2 / 2\sigma^2$$

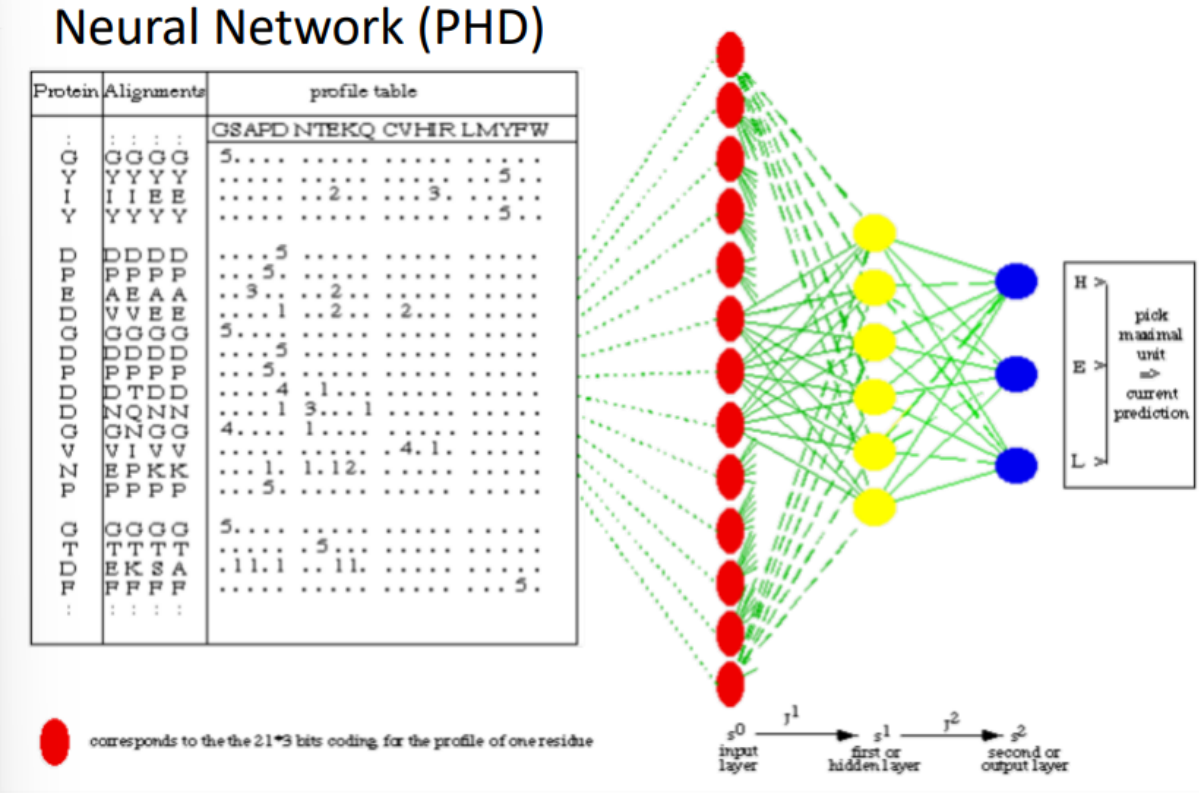
$$\frac{\partial \Phi}{\partial w_k} = -\sum_j \frac{(u_j - y_j)}{\sigma^2} \frac{e^{-x_j}}{1 - e^{-x_j}} V_k^j$$

Neural Network (add many perceptrons or neurons)
The protein sequence can be transformed into a set of vectors on the space of residues (dimension 20). Inputs can check by windows of 15 Aa along the sequence We can use more than one neuron, forming a layer of neurons. We can add multiple layers formed by neurons.



One thing to take into account about adding more neuron layers is that we are adding more parameters, so by increasing the parameters we are also increasing the probabilities of overfit, so we won't have enough data for all the parameters, getting a fantastic result for the training set and a terrible result for the testing test. The number of parameters is not unlimited.

PHD: Still one of the best Neural Network for prediction of secondary structure, which uses a matrix of multiple sequence alignment of sequences. Uses a window of 15 aa.
Second layer avoids errors so normalizes the results (imagine we have CCCHHEHHCCC, which it can not be possible since a helix needs 4 aa to do a turn, so that E must be an H and the second layer will avoid this type of errors).



3.2. Method of fold recognition TOPITS and THREADER

Let's go back to the original idea. We want to know the structure of the protein.

We just do a simple alignment between the predicted secondary structure and the secondary structure of the template, similar to a Needlesman and Wunsch.
If I have a good alignment, it means that the order in which we have the secondary structure being predicted is the same as the template.

One method is TOPITS and the other is THREADER

Fold prediction

But what happens if the fold is new?
So we need to create a new fold so ab initio fold prediction which is more difficult. There are different methods such as Rosetta, I-TASSER and AlphaFold

ROSETTA
We will first revisit the knowledge-based potentials and other things.
These methods were successful when we introduced "mutual information", which is an extra information for distance restraints.

1. **Revisiting the knowledge-based potential**

Very similar to the inverse of the Boltzmann law but using probabilities.

Now that we know the Bayes Theorem, we can calculate the probability of having a structure when I know the sequence.

$$P(structure \mid sequence) = P(structure) \times \frac{P(sequence \mid structure)}{P(sequence)}$$

We can simplify this equation. We can think of the sequence as 2 aa:

- The probability of the sequence of 2 residues → P(aai, aaj)
- Probability of the structure given the sequence → P(rij | aai,aaj)
  - The structure between 2 residues is the distance between the 2 residues.
- Probability of the structure → P(rij)

$$P(sequence \mid structure) = \prod_{i<j} P(aa_i, aa_j) \times \frac{P(r_{ij} \mid aa_i, aa_j)}{P(r_{ij})}$$

But the thing is that my sequence has multiple pairs of residues. So, to take them all into account, we do the product (if the pairs are independent).

Now, I have a term for the first equation.

So, I can calculate the probability of the structure given the sequence using:

- One factor that depends on the structure
- Probability of the distance between 2 aa divided by the probability of the distance

$$P(structure \mid sequence) \cong e^{-RG^2} \times \prod_{i<j} \frac{P(r_{ij} \mid aa_i, aa_j)}{P(r_{ij})}$$

This second term is exactly the same formula we were using the inverse of the Boltzmann law. P(rij) is for normalization.

Given the radius of gyration of a protein structure (RG), we approximate the probability that this is the structure for a given sequence, where the sequence is defined as the (foto de la siguiente página).

But when we have a full structure…

2. New potential based on condition probabilities

If we have a full structure, it is not only the distance it is also the environment in which the aa are (the system).
If I can calculate the whole probability of the sequence just using pairs of amino acids.

For a single amino acid, there is also a term of structure that is the environment. The residue is exposed or buried. If I have 2 residues, I will have to take into account what is the distance between them and also the environment:

- Are they both exposed
- Buried
- One buried the other exposed…

So, the statistical potential that we were calculating before should be split in different terms, depending whether the aa are exposed or buried.

We use this when we have a big protein that has aa buried and exposed. Otherwise we do not use it.

By applying Bayes theorem on a sequence (set of elements amino-acids), we can approach the conditional probability with respect to the structure in which the sequence is folded with the first two terms of the expansion:

$$P(x_1,x_2,x_3,\ldots,x_n) \cong \prod_i P(x_i) \times \prod_{i<j} \frac{P(x_i,x_j)}{P(x_i)P(x_j)} \cdots$$

$$P(sequence \mid structure) = P(aa_1,aa_2,\ldots aa_n \mid structure)$$

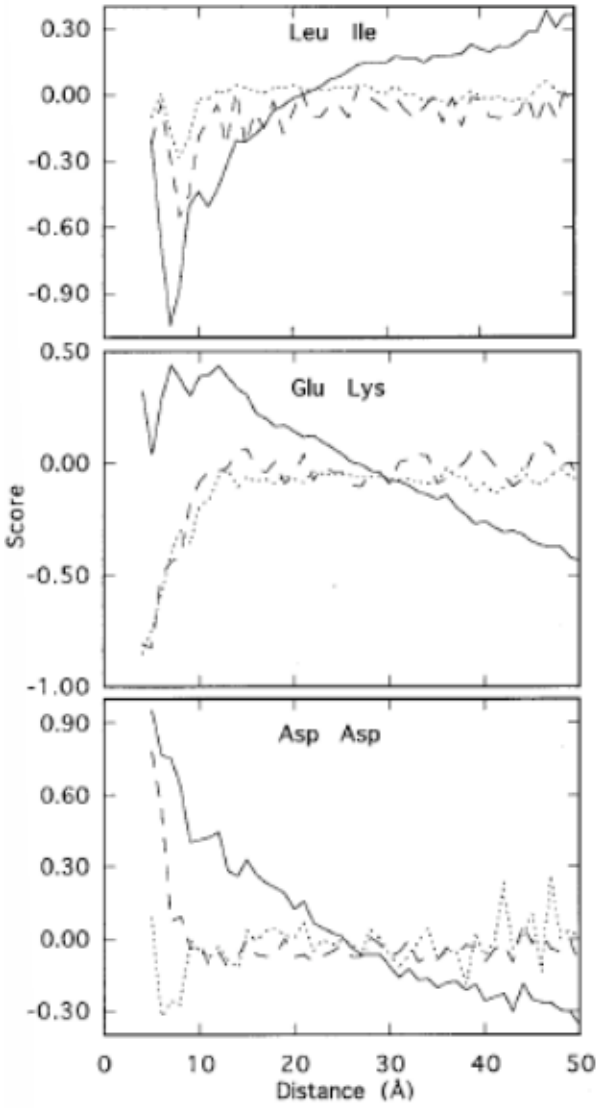
$$P(aa_1,aa_2,\ldots aa_n \mid structure) \cong \prod_i P(aa_i|E_i) \times \prod_{i<j} \frac{P(aa_i,aa_j|r_{ij},E_i,E_j)}{P(aa_i|r_{ij},E_i,E_j)P(aa_j|r_{ij},E_i,E_j)}$$

$$P(structure \mid sequence) \cong e^{-RG^2} \times P(aa_1,aa_2,\ldots aa_n \mid structure) \qquad \text{(Equation 2)}$$

Where E_i is the environment (secondary structure, accessibility, etc.) of residue aa_i

Let's see an example:
This is calculating the statistical potentials taking into account the environment.

- The results with a continuous line correspond to equation 1 (we do not use the environment)
- The results with a dotted line correspond to equation 2 (take into account the environment) for 2 buried residues
- The results with a dashed line correspond to equation 2 (take into account the environment) for 2 exposed residues



Sometimes, the effect is very small. For example, between Leu and Ile, the effect is not that big. Since both of them are hydrophobic, the behavior is similar.

But if we have residues that are charged (GLU and Lys or Asp and Asp). The behavior is different.

3. 9-Fragment DB of structures

Rosetta applies the following strategy:
Splits the sequence in fragments of 9 residues, using a window-like method.
Then it checks these fragments of 9 residues in a database of 9-residue fragments extracted from the total set of protein structures (split the whole PDB in fragments of 9 residues).
Meaning that it compares the obtained fragment against the whole database of fragments of 9 residues.

Rosetta assigns the first 25 most probable 9-fragment segments to a 9-residue fragment of the target sequence by selecting those with smallest score:
It is like a mini threading.

This mini threading compares the sequence and it also compares the score of the structure using the first equation.
It doesn't take into account the environment as the fragments are very small.

$$score = \sum_{i=1}^{9} \sum_{aa=1}^{20} |S(aa,i) - X(aa,i)|$$

Where S(aa,i) is the frequency of aa in position i of the target sequence and its homologs in the same 9-residues fragment. Similarly, X(aa,i) is the frequency of aa in position i for all similar 9-residue fragments (with the same structure).

So, it selects the best fragments and keeps going in a kind of sliding window of 9 residues.

At the end, we have several results:
25 fragments for each fragment of your target sequence.
Imagine that we have 10 fragments → 25^10 (it's too many)

So, among all the possible solutions, you select a few of them.
The union of these "solution" fragments gives as a result a certain fold.

Now we start using the second potential and get an improvement of the fold.
As it allows for several conformations, we have different solutions, and with the scoring function (second equation) we select the best one.

4. **Simulated annealing construction**

It is a kind of optimization with a MetropolisMontecarlo

Rosetta applies small changes in torsional angles for each fragment considered in order to join the 9-residue fragmented structures assigned to the 9-residue segment of the target

A conformation is selected according to the most probable structure-score: P(structure|sequence). A MetropolisMontecarlo simulation is applied using a simulated annealing

The structure-score is first calculated with equation 1, and when the simulation obtains a closer and more definite structure equation 2 (with more detailed potential) is applied.

iTASSER

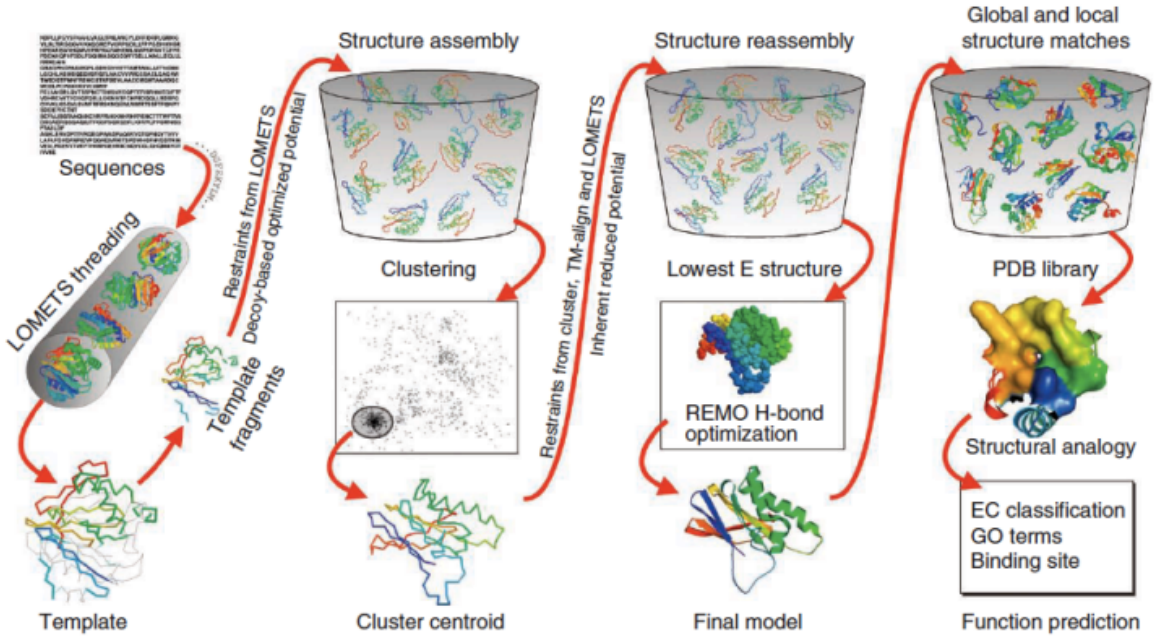
It is very similar to Rosetta, but in this case, instead of using fragments of 9 residues, it checks with HMM, FUGUE, LOMETS… and selects the best matches of small fragments.

Like running a blast and getting small matches of the target sequence from the PDB.
Some of this fragments are shorter or longer than 9 residues.

Then you do the same as we were doing with ROSETTA:

- Combining them
- Scoring them
- Then selecting clusters that have been seen many times when constructing the whole structure. Those clusters are normally the ones that contain the final structure.
- Then it optimizes the structure and gets the final structure.

iTASSER uses LOMETS threading. LOMETS uses the results of several threading approaches based on remote homology (i.e. FUGUE, HHSEARCH, etc.) and selects the common fragment-templates to assemble the target structure. Then it follows a similar approach to Rosetta.

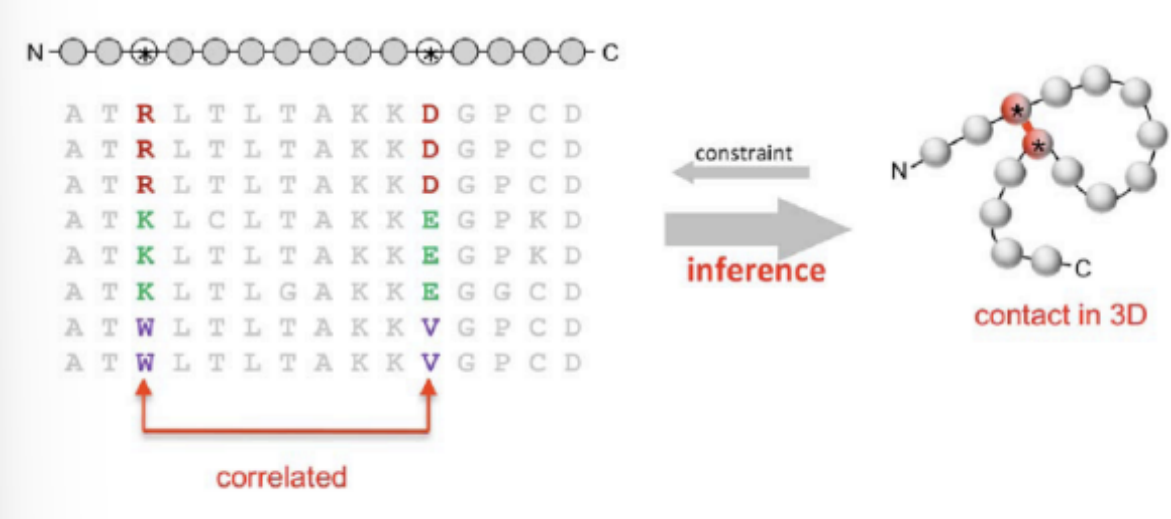


It was one of the best methods until Alpha Fold appears

5. Mutual Information

These methods started to get very successful when people introduced something called Mutual Information.

If you have the sequence of the protein, you may have 2 residues that make an interaction. Thus, these 2 residues that are connected in the 3d structure, will also be connected in evolution.
If along evolution, if one of the positions mutates appearing a residue that doesn't interact anymore with the other connected residue (should have different properties, otherwise the interaction will still exist), there will be a kind of pressure that will make the other (or the same) residue mutate into another in order to interact again.



So if this is true, those 2 positions are not independent. So the frequency of having those residues at those 2 positions will not be the product of each.

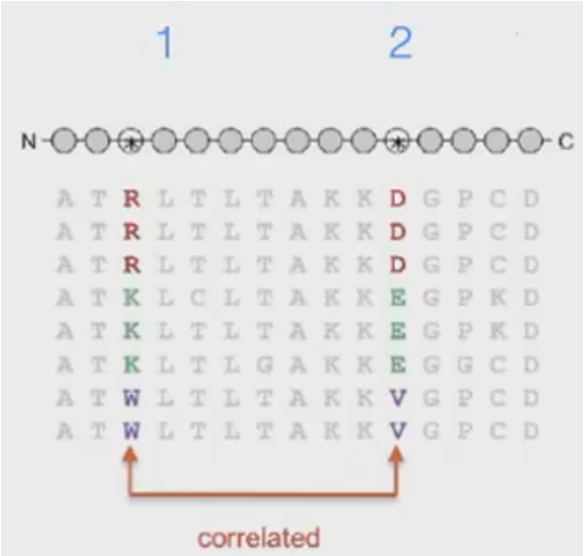
$$\mathrm{MI}_{ij} = \sum_{A,B} f_{ij}(A,B) \ \ln \frac{f_{ij}(A,B)}{f_i(A) f_j(B)}$$

Thus, in this formula for the Mutual Information, the ratio will be different to 1.
This formula (it's a kind of Shannon Entropy) takes into account what is the independence between these 2 positions:

- If the 2 positions are independent → ln(1) = 0 and therefore MI = 0
- If the 2 positions are dependent → MI can be very high

People used this MI to extract information on the possible interaction between residues.
The problem is that this approach didn't work.

Explanation why it did not work:
If you have position 1 correlated with position 2 and they are interacting.



What happens now if we add a third residue that interacts with the second residue?
Using the formula of the MI, we will consider that:

- Since 1 interacts with 2 and 2 interacts with 3, 3 also interacts with 1. Which is false!
- This is due to the fact that the MI formula is transitive

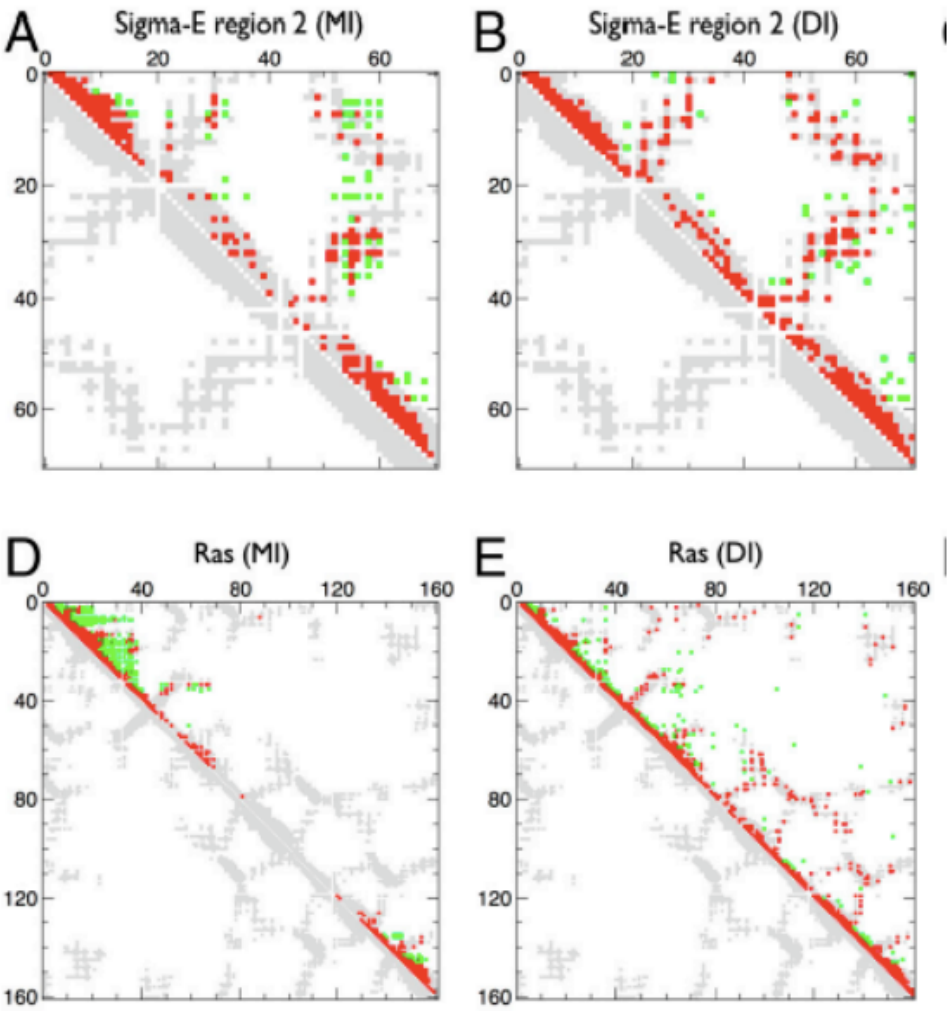
This was a problem until we started using some new equations → Direct Information
Which only takes into account the interaction between pairs and not triplets…
So, it does not have an effect from this transition relationship that MI has.

Example:

- Green → Long distances
- Red → Short distances

A plot uses (MI) and B plot uses DI → In both cases we have a good result. So, we would be able to solve the structure of the protein.

D plot uses MI and we have a bad result → There is a concentration of points at the beginning.
E plot uses DI and we have a good result because it spreads the values.



What would happen if we add this information to iTASSER or Rosetta?
This would improve the results of ROSETTA and iTASSER.
It's not just the local regions of the different fragments but also how to place those fragments with a restriction in the 3d space.

This is very good in transmembrane proteins.

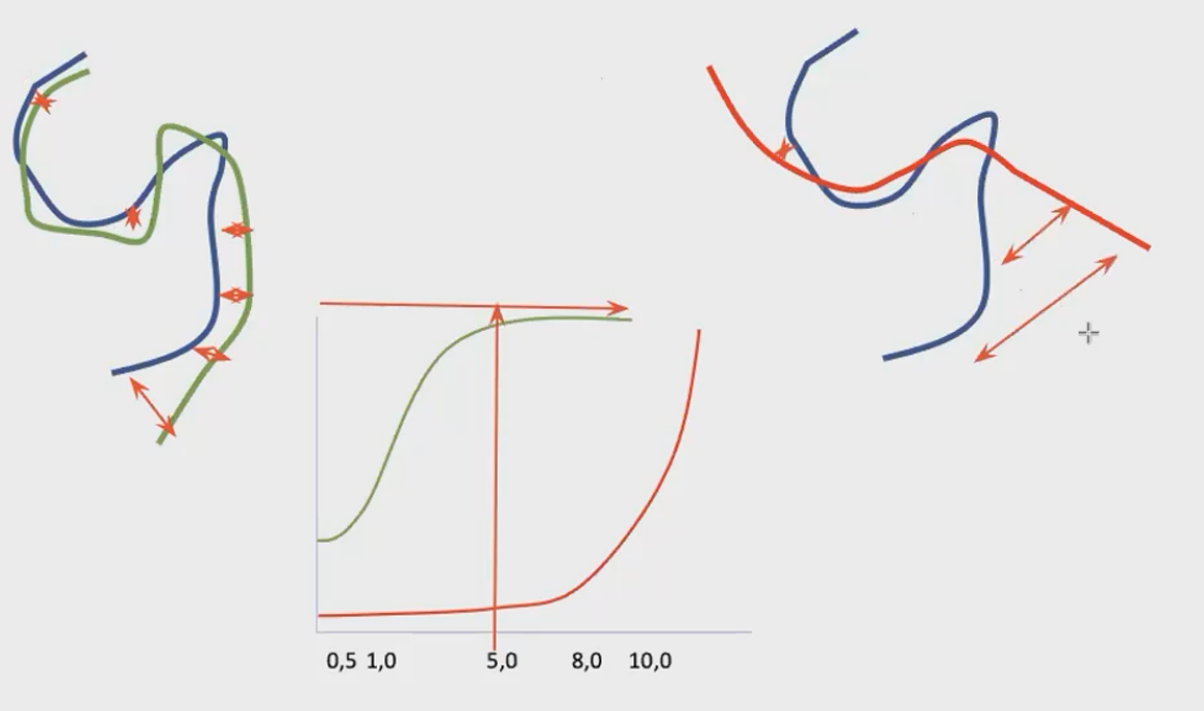
CASP (Critical Assessment of Structure Prediction)
Competition to make a prediction in 2 years time.

- Input sequence
- Output structure

We need to compare the structure obtained with the real structure. So we need an evaluation.

- Use the GDT score → Calculated as the largest set of aa residues (alpha carbon atoms) in the model structure falling within a defined distance cutoff of their position in the experimental structure, after superimposing both structures.
- We calculate 20 different GDT scores using 20 different cutoffs (0.5A to 10A).

In the following plot, we can see that we obtain the green structure with less than 5A.
The red structure is very bad.



The conventional GDT_TS total score in CASP is the average result of cutoffs at 1, 2, 4, and 8A. So, I get a single value.

We get the % of aa that is at his distance.

The larger value of GDT_TS the better

There is another evaluation called **TM_score**, which is based on a program called TMalign.
TM_score is a measure of similarity between 2 protein structures.

- TM_score = 1 → Perfect match
- TM_scores > 0.5 →Roughly the same fold
- TM_scores < 0.2 → randomly chosen unrelated proteins

Docking and macromolecular complexes

Molecular recognition

All macromolecules work through recognition processes.

- Protein-ligand.
    - Enzymes, membrane receptors, transport proteins, Drugs
- Protein-protein
    - Regulation of enzyme activity (signal transduction), multi-subunit protein and complexes
- Protein-NA
    - NA metabolism, gene regulation
- NA-Ligand
    - Drugs
- NA-NA
    - Transcription, replication, protein synthesis, ...

- Recognition is selective. It depends on the participating groups.
- Recognition is dynamic
    - Induced fit / Conformational selection
    - Complexes can be permanent, but most of them are transient.

Energy considerations

Entropic

- Conformational, hydrophobic

Enthalpic

- Vdw
    - Shape and contacts
- Hbond
    - Define geometry, must be complete
- Electrostatic
    - Severe solvation penalty

Structural complementarity: We are going to use this complementarity as a prediction method.

Complex formation implies to bury new interactions

- Unstable (hydrophobic) surfaces in water may indicate binding regions.

Proteins do not act alone

Protein – protein complexes:

- Permanent associations (“quaternary structure”). They are the easiest to predict.
- Enzyme – substrate (transient). Really difficult to predict because it is not stable enough.
- Regulatory associations
- Multi protein clusters/groups
    - (Nuclear porus, cytochroms, …)

Protein – NA complexes

- Transcription factors
- Replication, Splicing, Transcription machineries
- Ribosomes

Concepts and definitions

Docking: Prediction of the structure of complexes

- Ligand-protein docking, protein docking

Receptor, ligand: The actors. If they are both 2 proteins, how can we define which is the receptor and which is the ligand? The smallest one is the ligand because docking works by taking the ligand and moving it through the receptor. So, by taking the smallest protein, we will make less computations.

Pose: Binding mode

- how the ligand positions on the receptor site

Scoring: How the binding is evaluated (tries to approach to the binding energy)

(Virtual | Reverse) screening: Test for the feasibility of binding among a high number of ligands/receptors. Select just a few that are really good.

Docking evaluation: How to test the success of the docking

- Protein docking is really bad

Protein-Protein docking terms

There are 3 levels:

- Interaction, interface prediction and protein-protein docking.

The simplest thing to do is to predict the **interaction**, not the complex.

- Predict whether 2 protein interact with each other
- This is normally the only information we want. For example, if we want to know which are the proteins that regulate a metabolic pathway, we just need to know if 2 proteins interact, not how the whole complex is formed.

Then we can make an **interface** prediction.

- Determine the regions where complex components interact. Prediction of the binding site. So, we just want to know which are the residues of the proteins that are interacting.
- This is important because, imagine that we want to design a drug to cancel a complex, I need to know that our inhibitor binds in that interface region.
  Or just do mutations in that region.

Protein-protein docking.

- Predict the structure of the whole complex.

Bound vs. Unbound docking (harder)

- Docking using conformations in the complex (bound) or free (unbound)
  - Bound → When the structures of both proteins are known (taken from PDB)
  - Unbound → Structures of the individual proteins or their specific conformation within the complex are not known.
- It is different because the structure of the unbound forms are different to the bound forms because there are small adjustments when complexes are formed.

Flexible (harder but better) vs. Rigid docking

- Whether proteins' flexibility is taken into account. We allow the proteins to change their conformation.

Local vs. global docking

- Whether binding site is roughly known
- We use local docking when we have information about the binding site. Much better.

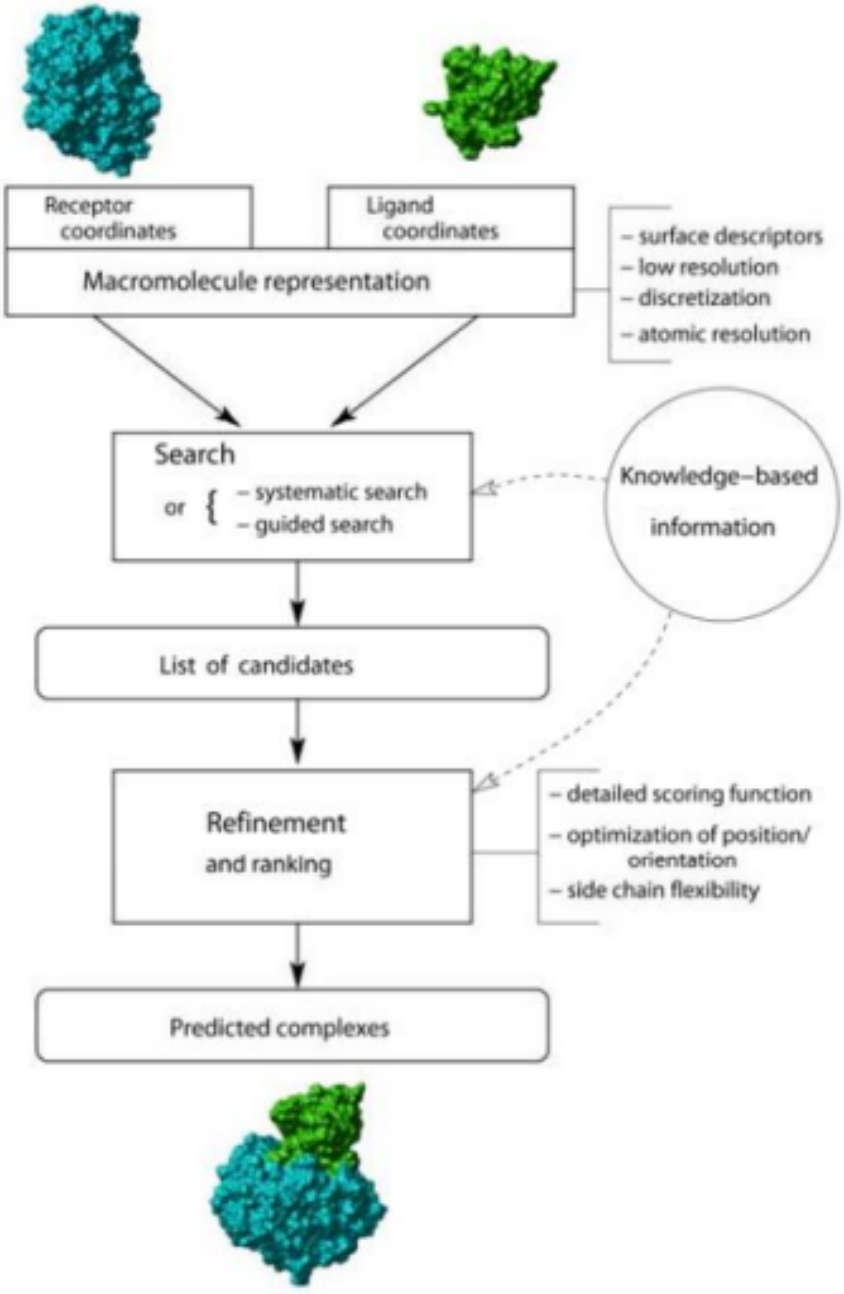
Docking strategy

1. Protein representation
2. Search method
3. Scoring method
4. Refinement

We start with a 3D structure and we search the binding site by moving one protein around the other until it fits. This is called systematic search.

If we do know where the binding site is, we use a guided search.

Then, we have a list of candidates (poses) and we do some refinement and a ranking.



So, as we have seen, there are some things we need to decide:

- How do we represent the protein
- Which are the search methods → Systematic or guided
- Which is the scoring method → Later
- We do a refinement or not → Normally not

Ligand-Protein Docking

It is the most developed type of docking and it normally works.

Molecular Docking: It is the closest thing to 3D structure prediction. So, we want to predict the 3D structure of the complex.

- Prediction of 3D structure of ligand-protein or protein-protein complexes. We want to obtain a new PDB with the ligand in the correct position.
- We need this level of detail because this is the kind of structure that we are going to use to evaluate binding energies.
  So, it is going to decide if a drug is better than another, for example.
  - **Quality of the structure is the main objective.** Realistic binding energies
- Usually combined with other techniques, as MD.
- Experimental information can be considered

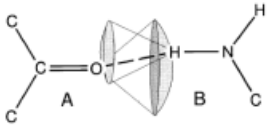
Virtual screening

- Identification of a few possible ligands from compound databases
- We select one receptor and multiple ligands (> 10^6)
- Calculation should be fast (>10000 ligand-receptor dockings / day / proc.)
- The main objective is to select "some" ligands, that can be optimized with other methods

Reverse Screening: Once we have optimized that drug so that you have 3 big candidates, we can check if these candidates apart from blocking the protein of interest, it also blocks other regions.

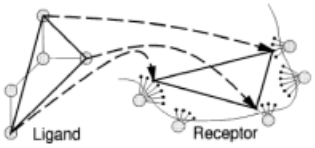
- Identification of possible receptors for a known ligand
- One ligand - multiple receptors
- Points to possible side effects

Representation of Ligand-Protein Docking



Complete atomic representation

- Large cost, high resolution

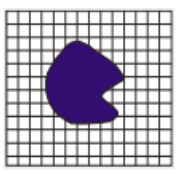


Simplified representations: For visual screening

- Quick and robust. Low resolution.

3D Grid representations. Map the receptor in a grid

- Easier energy calculation
- Definition of "pharmacophores" (MIPs)



Flexibility (Ligand and receptor)

- Ensembl docking

Does a rigid molecule bind better than a flexible molecule?
Rigid molecules. Because flexible molecules must choose a conformation and then bind (they lose entropy). Rigid molecules do not need to choose a conformation, they fit or not.

So, flexible molecules are generally bad drugs.

From the point of view of Docking, ligands are not that flexible (because we choose molecules that are not that flexible). So, most of the methods use some approach to take into account ligand flexibility.
The easiest thing is to do ensemble dockings → Take a bunch of conformations and try all of them.

We nearly never include the flexibility of the receptor, because it is too big.

Scoring

Recap → We start searching structures, we decide the representations, decide the search method (systematic search in ligand dockings is almost never needed, because we know where to find the active site. The opposite happens in protein-protein docking) and then we make the score.

We optimize with a scoring system → We move the 2 proteins until we get the best analysis. So, we take all the possible orientations of both proteins and evaluate the binding energy as we did in biophysics.

This works, but it's not efficient.

When to score (it's the same for ligand or protein):

- Score associated to the search process
- Scoring a posteriori is always better. Reason → There are many ways of scoring and we want to test several methods with a list of candidates. The normal thing to do, especially in protein docking, is to generate all the poses and then order all the poses according to different methods.
  Then take a consensus decision of which is the ideal complex.

Structural complementarity is used to make the scoring.

- Robust, low resolution

Classical force-fields, Statistical Potentials are also used to make the scoring (better but it is computationally more expensive)

- High resolution
- Easy to transfer

Empirical functions are another way of scoring used in virtual screening (bad representations and bad scoring systems, because quality is not that important, you just want to go fast). They are too bad for molecular docking.

- Delta G bind obtained from function fitted to experimental data

Protein-Protein Docking

The same as before, but everything is more difficult and nobody pays for it… No-one is interested in selling software that does this type of docking.

However, from the point of view of biochemistry, this is far more important. From the point of view of drug design, now that we are starting to know complexes much better, some people are starting to build drugs to avoid the formation of complexes. So, instead of inhibiting an enzyme, we inhibit the formation of the complex that activates that enzyme. Meaning that we have to build a drug that goes in the interface between 2 proteins.

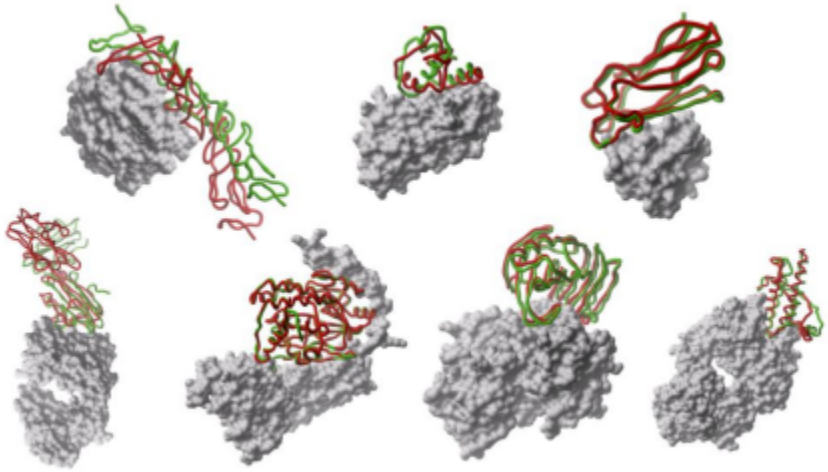
This is much more difficult because the interface is much larger.

- Typical contact area: 1,500-3,000 Å^2

Meaning that there are many degrees of freedom of putting these 2 surfaces together and a much bigger influence of the conformational flexibility (drugs are rigid but proteins move a lot). Just a single 2 Å movement of the side chain may kill a docking prediction.

The easiest energies to calculate are the polar contacts (electrostatics, H-bonds…):

- Most of the protein-protein contacts are hydrophobic, which depend on the entropy of the solvent and we can not calculate too much about the hydrophobicity.
- There is a very low number of H-bonds: Average 1 H-bond / 170 Å^2 (*)
- There is small number of water molecules: 1 water / 100 Å^2 (*)



As we can see, the complexes differ a lot. They can have a big/small interface, different locations…

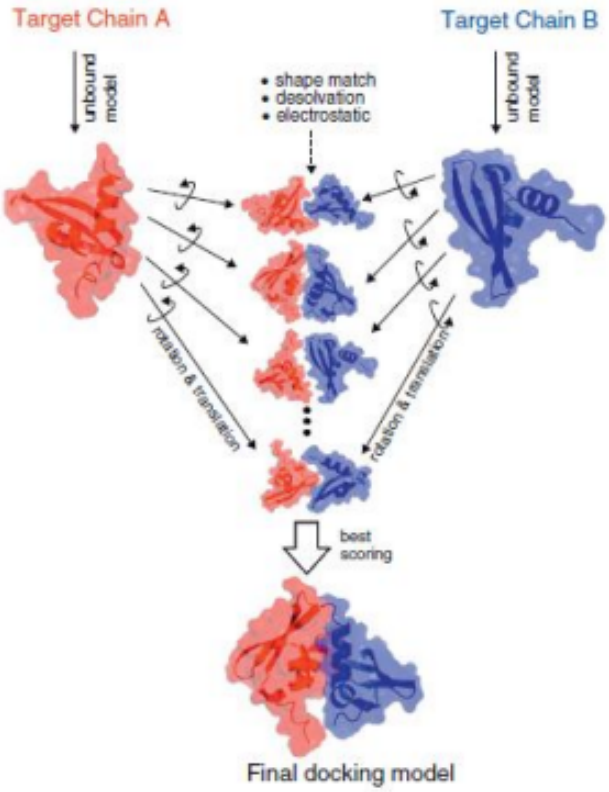
So, we will need different methods or strategies for each case.

2 Main Basic Strategies

"Pure" Ab Initio docking

It is similar to Ab Initio 3D structure prediction, in which we start with a sequence and it folds magically. In our case, we start already with the structures, so its better than Ab Initio:

- Only information about ligand and receptor structures is known:
  - You get the structure of one chain, the structure of the other chain and start playing with it, creating a number of complexes. At the end, you select the best scoring ones.



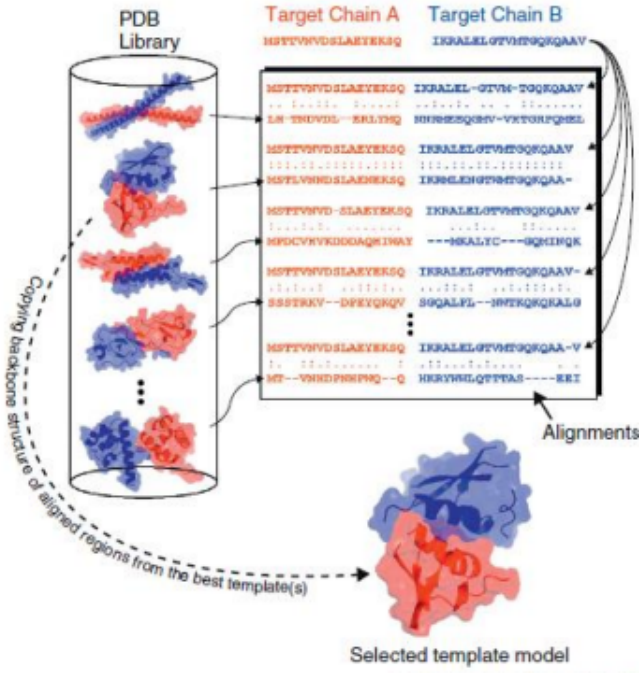
- The methods used:
  - Pseudo-random approaches (simulation, optimization) → Make all the possibilities and choose the best one
  - Directed search (Geometric hashing) → Try to match the shape of the 2 proteins and then create the complex.
  - Brute-Force approaches (Grid-based, FFT) → This is the most popular.

Data driven docking (template based)

Homology modeling of fold recognition → We copy from existing complexes.

- Experimental, homology data is used. So, you have a lot of sequences from each chain and you do a MSA (typical things we do in homology modeling) and then we take from the PDB some templates using the sequences.

- The methods used:
  - Experimental, homology data → Most popular
  - Machine-learning methods
  - Co-evolution methods
  - Can be combined to help Ab Initio approaches

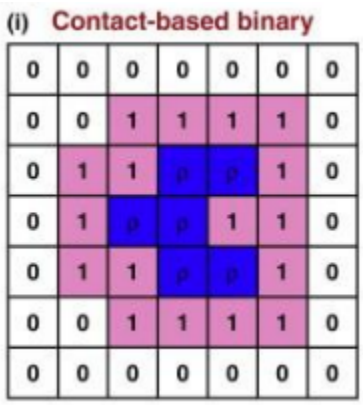


This second alternative is much better. But sometimes, you can not do homology modeling (you have no sequences) and you have to do Ab Initio.
There are really high chances that you find an homologous sequence in the PDB, because proteins are not that different. But in our case, we need to obtain 2 homologous chains and then find a PDB that has both of them, so it's a bit more complicated.

For this reason, Ab Initio docking is much more popular than in 3D structure prediction.

Ab initio Rigid-Body docking
The idea is very simple:

- Proteins are mapped onto 3D grids. Each grid point is evaluated as inner (blue), surface (pink), outer (white).



- Blind 6Dim (3 translations x 3 rotations) search
- **Score is based in 3D complementarity**: i.e. matches among the “Surface” points (Calculated with fast bit-wise arithmetics)
- Fast Fourier Transforms to speed up translational or rotational (Spherical Polar Fourier, SPF) searches. To do this efficiently, we need to do all the possible translations and rotations, so it's a systematic search.
- Computational cost can decrease by >10^4 (from N^6 to N^3 lnN^3)

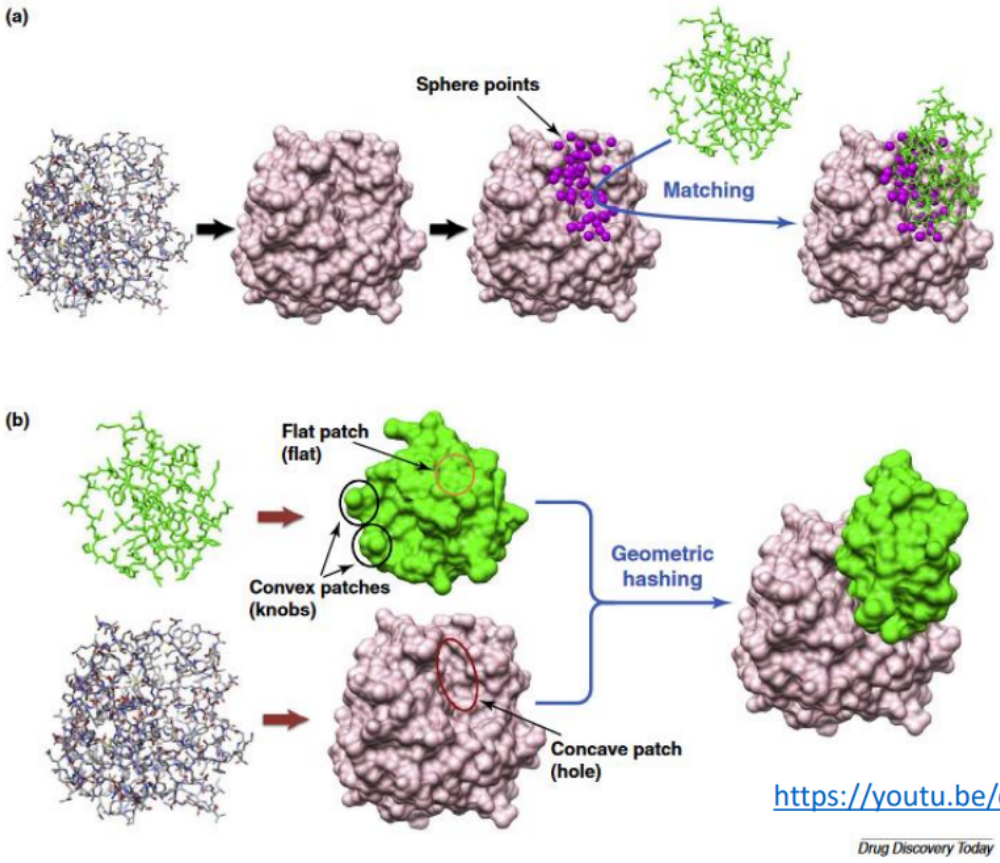
Since we are looking for 3D complementarity of the chains, the complexes with larger buried interfaces are easier to predict.

There are other approaches like:

- **Geometric hash**, Surface are pre-processed to detect possible matching regions (like making a puzzle).
  So, we kind of generate spheres around the surface of both proteins and then we match them.

  So, in practical terms we just try to find concave and convex patches that can be joined, forming the complex.

- Solvation/desolvation can be mapped into Surface properties
- Most favorable poses are re-scored with better scoring functions. For example, if both surfaces have opposite charges. Or they are both polar or hydrophobic.
- Less efficient for pure blind docking (better with additional information).



Additional tricks

Things that we add to improve the docking.

For flexible docking

- Traditional MD. This is theoretical the best approach, because we are adding flexibility.
    - Too costly (limited sampling)
    - For this reason, it is used to refine structures after docking. Sometimes it is also used to relax the side chains that are in contact in the interface.

- Conformational sampling
    - Rigid docking with set of posible conformations (experimental or produced from PCA)
    - Conformational search added to position and orientation (usually Monte Carlo). So, apart from doing 3 translations multiplied by 3 rotations, we will also have to multiply for N possible conformations.

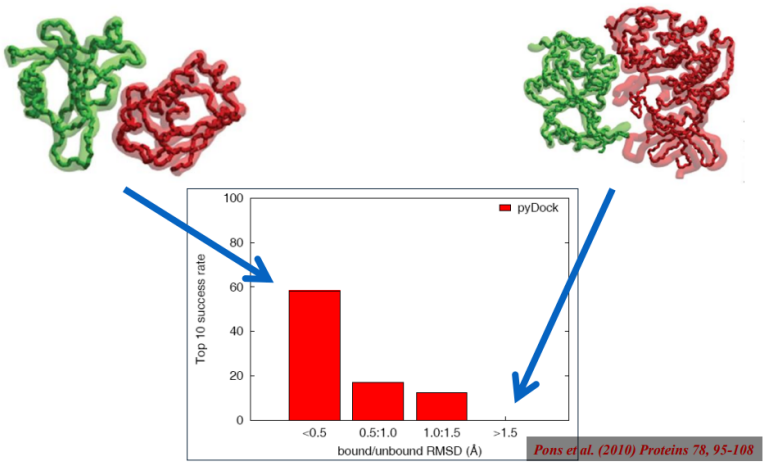
- Combined cycles of docking and then simulation to relax the system.
- RosettaDock combines rigid body MonteCarlo for orientation/translation + MonteCarlo among rotamer libraries (very expensive)

Soft docking

- Backbone is still rigid but the Sidechain flexibility is mimicked using “soft” VdW potentials, and/or coarser FFT grids (allowing small collisions, meaning that we make the VdW radius smaller)

Rigid-Body limitations

Flexibility is the worst problem in docking and it is the reason why most of the protein docking methods fail.
If a protein is to flexible, the difference between the 2 monomers separated and the complex will be big.



If we have additional information, we can add constraints on distance.

Data driven methods

Homology/threading based methods (same concept as we have seen with Baldo)

- Template based, use data from homologues

Co_evolution methods

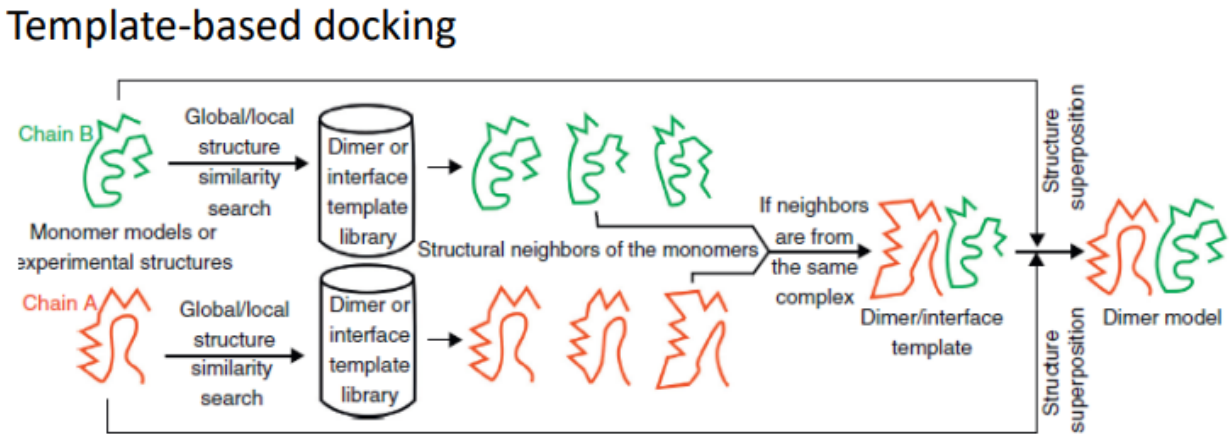
- Growing popularity in protein structure prediction
- Uses data from “massive” multiple sequences alignment

Interface prediction

- To reveal interfaces without structure prediction

This is a summary of first level of data driven methods:

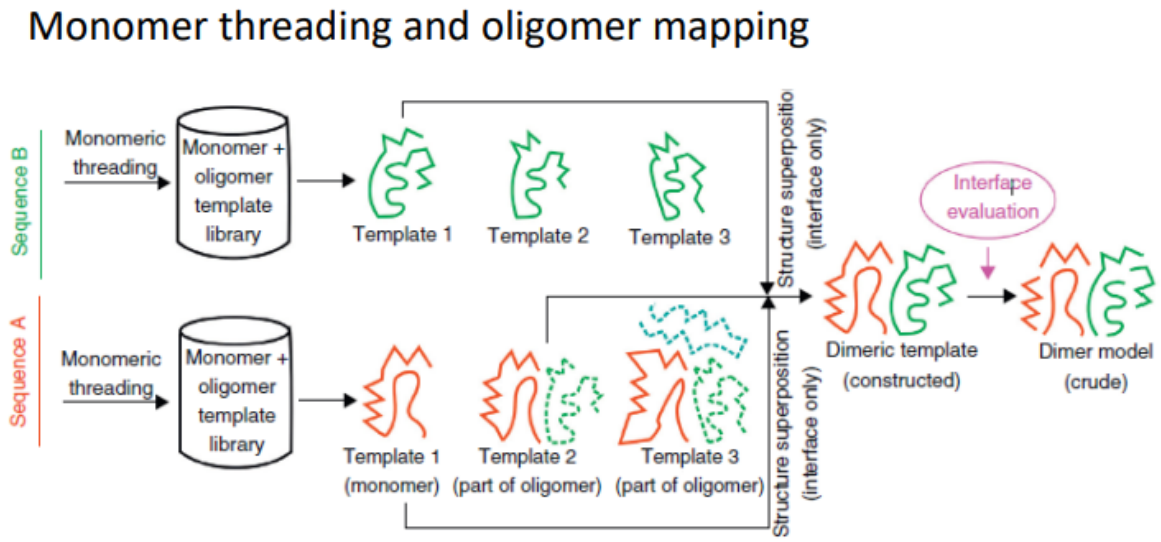
Let’s start with the simplest one, template-based docking (gives best results).



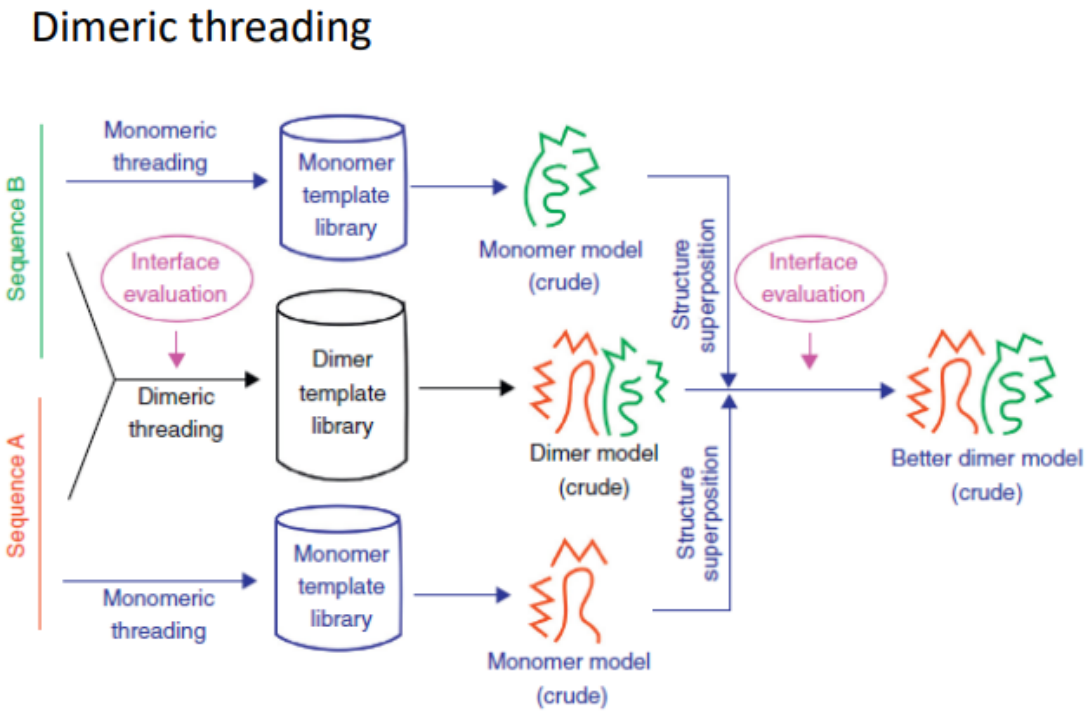
Take both chains and map them into the PDB and find if any of the homologous proteins to our targets A and B make a complex. If it's the case, we use the standard 3D structure homology modeling to get the structure of the dockings and then we superimpose our models into this complex.
Then you can do optimization…

What happens if you are not that lucky? We can just do threading. There are 2 levels of threading:

- This is similar to the fold recognition we did last week. So, we have 2 monomers and 2 libraries of templates (library of shapes) and then we try to do a docking with the predictions and we evaluate which are the interfaces, contact points and possible complexes you can make.
  In this case we have both structures at the beginning, but we could also do it with just sequences.



Dimeric threading → We do threading of the monomers but also of the interface of the model, so we need a third library that contains the dimer templates (complexes).
So, we have some guidance from this third library.

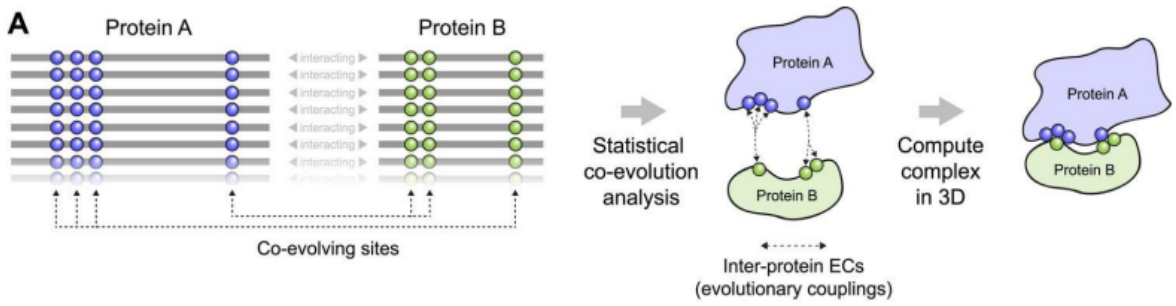


Co-evolution methods

We have 2 proteins, we do MSA and we end up with some statistical conclusion that some position in one protein and another position in another protein are coevolving. Meaning that there is a statistical significance that these positions are not mutating at random, they are related.

Same concept within 2 chains of the same protein.

- Rely in multiple sequence alignments and correlated mutations to define correlated positions > 3D contacts
- Predicted contacts become restrains

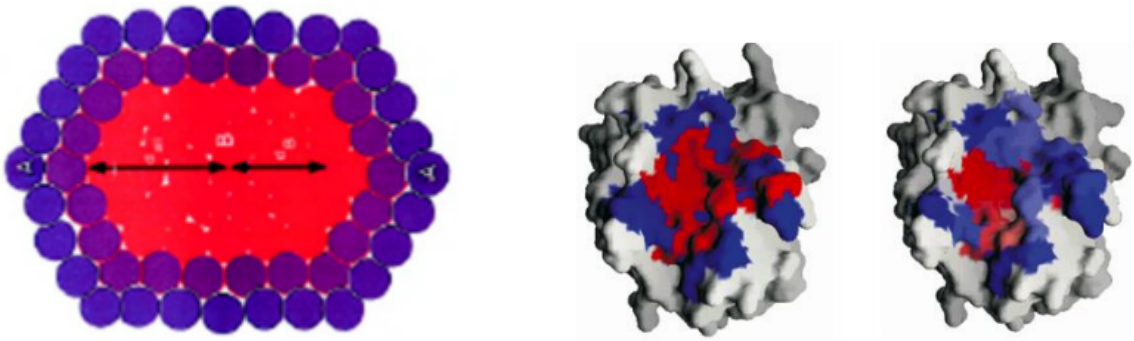


With this information and an enough number of sequences, you may derive distance restraints (you know the interface). So, we can restrict any of the methods that we will use to make the docking.

Interface/hot-spot prediction

Prediction of interfaces is important because many of the times we do not need the whole complex.

The hot-spot/O-ring model is the mental model of the protein interface.



The interface has 2 big regions:

- Hot-spots make the real interactions (red)
- O-Ring (residues surrounding the hot-spot) help to exclude solvent

Detecting solvent excluding (hydrophobic) regions help to detect interfaces

Residue packing at the interface is similar to protein core.
Residue conservation does not help (except for specific interactions)

- Amino acid composition
- Interface propensities (statistical models)
- Machine learning approaches

Definition of interface residues on training structures is key

- Geometric (Distance based)
- Optimal Docking Area (ODA), based in desolvation energy (ASA). We start in a hotspot and we extend the surface until the score drops (no more interface).

Good docking criteria

How do we decide if protein docking is correct? This is completely different to ligand docking. In ligand docking, you want a complex that is like the one you would get from the PDB, as good as that.
In protein docking it's impossible to go to that level. So, we are really happy to have:

- Low Free Energy
- Low pseudo-energy (low scoring value)
- Large Surface burial (~ 1,600 ± 400 Å2)
- Low VdW overlaps
- No large cavities on the interface
- Good H-bonding (~1HB / 100 Å2), Polar-polar contacts
- Good charge complementarity

Scoring functions

- Free energy (Forcefields)
- Solvation Score very important because we have a lot of hydrophobic contacts
    - Optimize hydrophobic solvation
- Statistical potentials
- Geometric scores that evaluates:
    - Buried surface
    - Surface shape complementarity
    - Volume of intersection
- Phylogenetic scores
    - Based on conservation
    - Correlated mutations - >contacts
- Re-scoring and consensus

Conclusions

Protein docking works

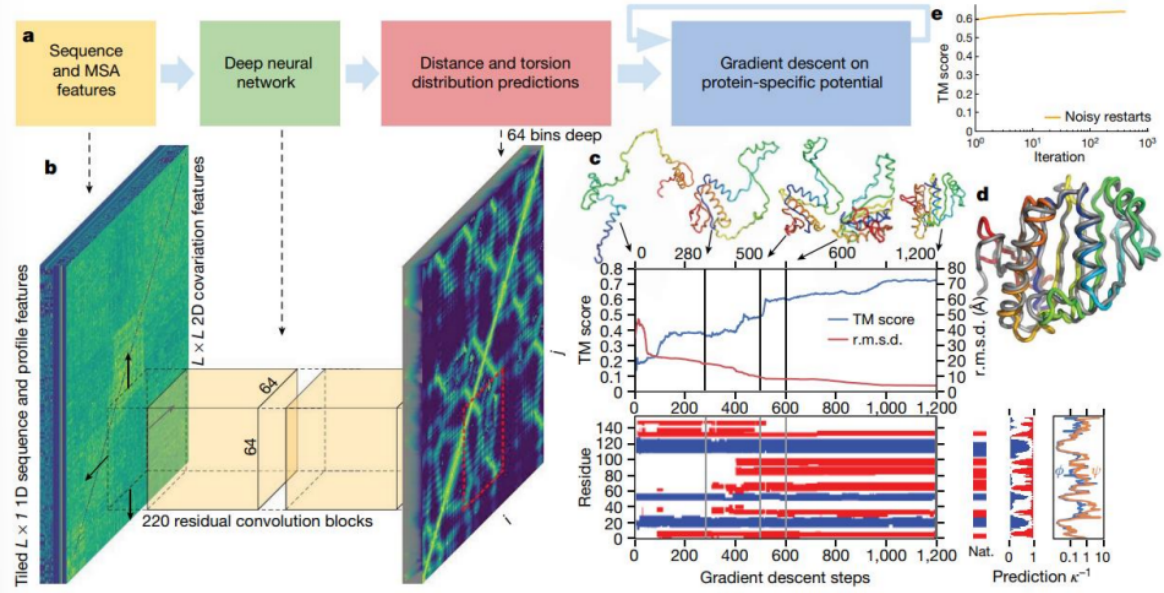
- (Much less efficient than ligand docking)

Lots of methods exists, no clear winner
Data-driven methods can generate better models if data is available
Flexibility, conformational changes are the major problems
Interface and interaction predictions (without docking) are possible and useful

Alphafold

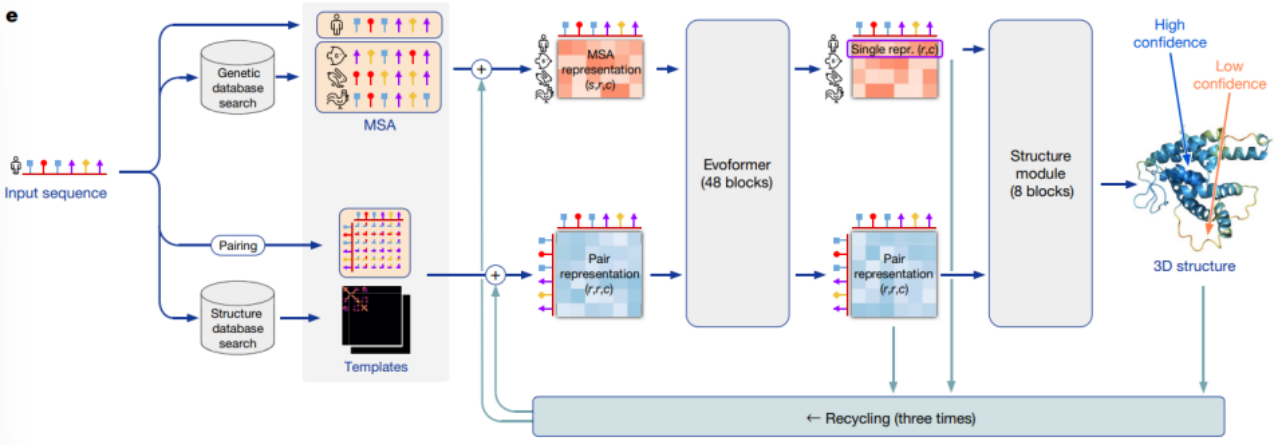
It takes as input an amino acid sequence and it predicts it's entire protein structure.

- First of all, it makes a MSA of homologous sequences of our target.
- This MSA is the input for the ML pipeline.
    - Information is going to be extracted from this sequence alignment and it is going to be transformed into a matrix of distances between pairs of amino acids.
    - This distant matrix is going to be used to reconstruct the model.



- In Alphafold 2 works in a similar way. It also starts with a MSA then we have a block of ML that is divided in 2 Blocks:
    - **Block 1:** Devoted to analyze and manipulate the sequences from the MSA
    - **Block 2:** Deals with the structure.

    So, we are going to extract the features from the sequences and we are going to represent them into terms of structure.



See that there is a recycling part that has an iterative behavior. So, the results that we get at different points of the workflow are going back and are used to perform a second or third step of refinement using ML.

What is happening in the 2 boxes?
Deep neural networks

Understanding machine learning

Machine learning consists of developing algorithms that are able to learn by themselves in order to improve performance at some task.

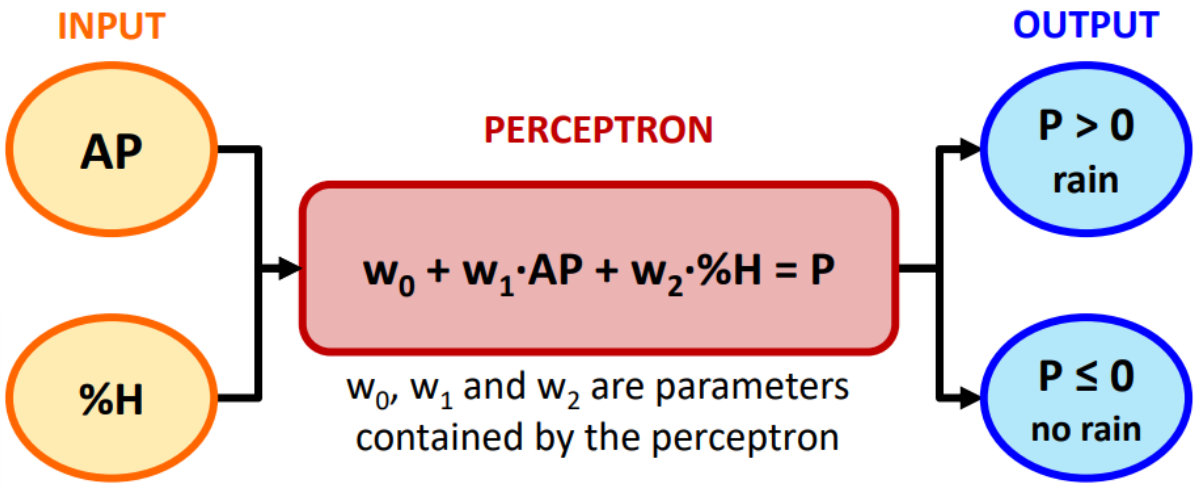
In the case of alphafold, the task is to predict protein structure from sequence using deep neural networks.

To better understand ML, we are going to understand one of the first ML algorithms ever: The perceptron

The perceptron is an algorithm that takes an input and with that is able to make a binary prediction (something happens or not). For example, we could use a perceptron to predict if tomorrow will rain using as input the atmospheric pressure and the humidity percentage of the day before.

The perceptron needs some inputs (Atmospheric pressure and % humidity) in order to do the prediction. So, it is going to apply a mathematical operation.

Inside the perceptron we have several parameters w0, w1, w2 → number of variables + 1

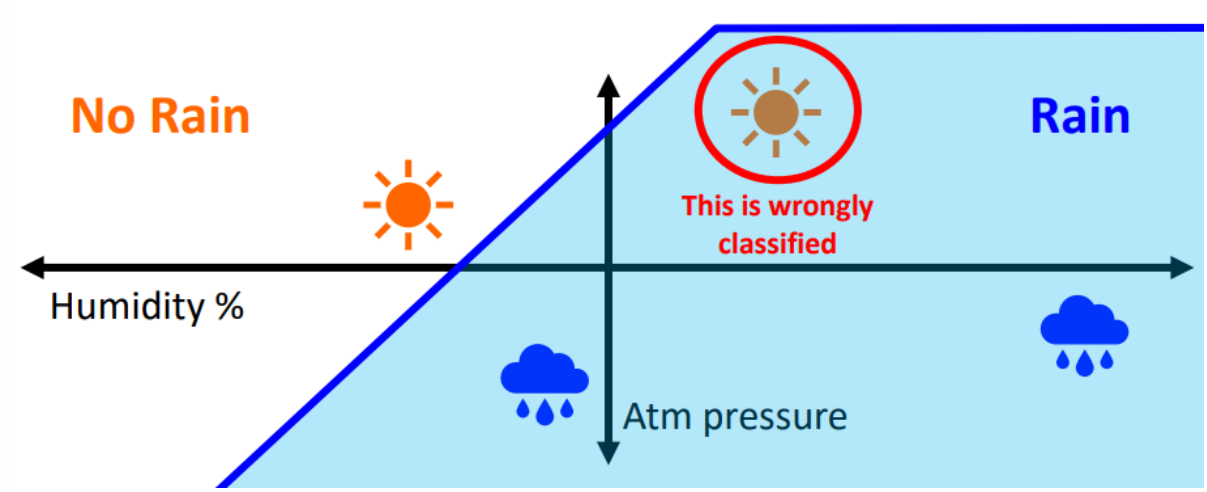


If P is bigger than a threshold, 0 in this case, it is going to rain.

The guys that came up with this idea were inspired by neurons. A neuron is a cell that receives some inputs and depending on the inputs it is going to fire or do nothing.
This is way the perceptron can also be called a model for an artificial neuron.

For a ML algorithm to learn, you need to train the algorithm. Meaning that you need to give some data to the algorithm so that it can improve.

Day	Atmospheric pressure	Humidity percentage	Weather the day after
Day 1	0.96 atm	70%	Rain
Day 2	0.93 atm	40%	Rain
Day 3	1.1 atm	30%	No rain
Day 4	1.23 atm	50%	No rain



To avoid these wrong answers, we are going to perform multiple iterations and in each iteration we are going to update the parameters that the perceptron has.

For each day that is wrongly classified, the perceptron updates its parameters (w0 , w1 and w2) by applying the next mathematical formula:

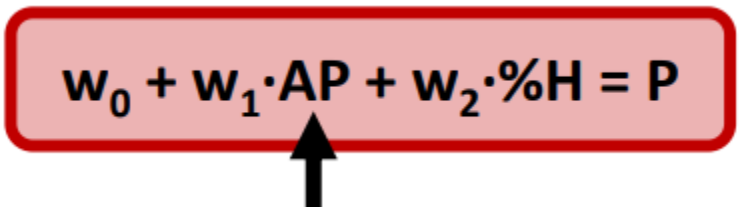
In this example we are going to update w1, but the same procedure applies to all the parameters in the perceptron:

$$W_{1,new} = W_{1,old} + LR \cdot D \cdot X_1$$

LR is the learning rate. It represents how fast the change of the parameters is going to be. If we have really high rates, the algorithm is going to struggle to find an optimal position, because it will change in big steps.

D takes into consideration the sign!
If the prediction was too high, D = -1 otherwise D = 1

X is the value of the misclassified day that is multiplying for the parameter that we are updating. In this case, the parameter that multiplies w1 is atmospheric pressure.



Now we execute the perceptron again with the new parameters.

From the perceptron to deep neural networks

One perceptron → One more complex neuron → Many neurons organized in a network → Networks of complex architecture (deep learning starts).

Hands on AlphaFold

All the results can be found in a folder because obtaining them takes a lot of time (we won't have to do this in the exam).

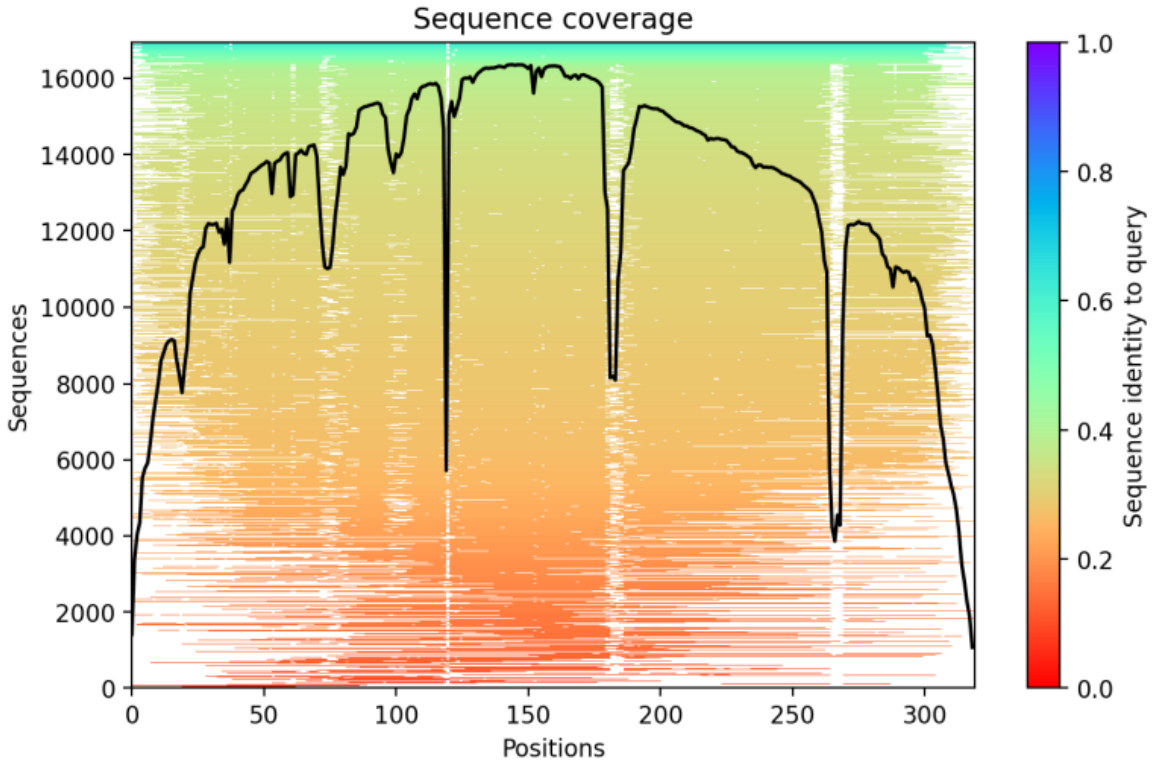
Moreover, AlphaFold2 has modeled most of the proteins in uniprot and shared the models in the following DataBase: https://alphafold.ebi.ac.uk/

AlphaFold2 will return us 5 models instead of 1. This is interesting because each model is slightly different to the others and therefore it is useful to see the structural variability that you can have in your target protein.

Step 1. Modeling a globular protein (we did this using modeller in practical 4 and 5 and we did not obtain optimal results)

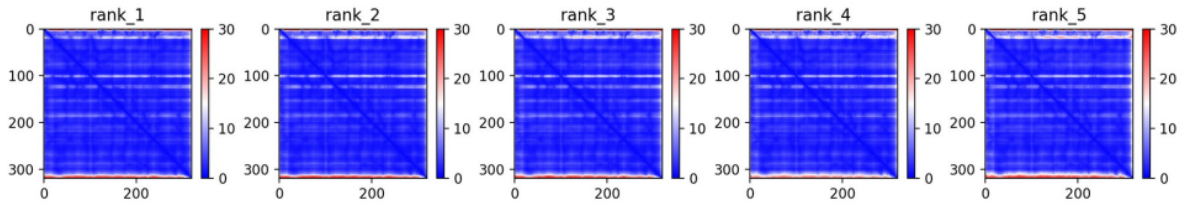
We have a lot of files, but we just need to focus on 5 PDBs and the plots.
AlphaFold apart from giving great predictions, it also tells you which predictions are reliable and which are not.

Coverage plot:



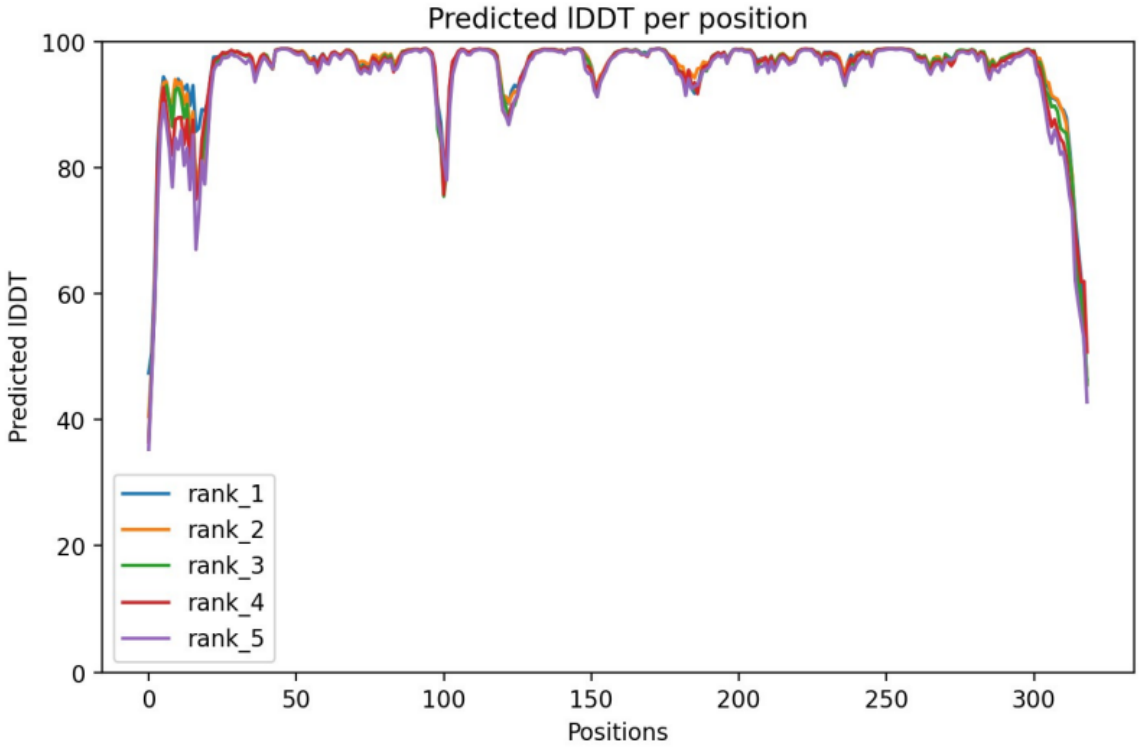
As you remember, the input for alphafold2 is a MSA. In this plot we can see the representation of all the sequences that have been used to create this alignment. The X axis represents the amino acid positions in our protein of interest, while the Y axis indicates the number of homologous sequences found for our target. See that some regions of our target have more coverage than others, as it is represented by the black line in the plot. Finally, the color of the sequences indicates the sequence identity percentage between the homolog and our target. We see that the coverage is quite high for most of the protein, with some exceptions in short regions (if you check these short regions in pymol, you will see that most of these short regions are loops).

Predicted align error plots:



See that we have 5 plots, where each one of the plots corresponds to one of the models provided by collab fold. In these plots alphafold is showing the predicted aligned error (PAE). The X and the Y axis represent both the amino acid sequence of our target protein, and the color is indicating the predicted aligned error. This predicted aligned error is a value that says how sure is alphafold of the distance between two amino acids. In this case, the PAE is very low for the entire protein, meaning that alphafold is confident for the prediction of this model. In the next executions of alphafold we will see more complicated cases where these PAE plots become more tricky.

Predicted LDDT per position:



Another metric that alphafold uses to quantify the quality of their models is the predicted LDDT (Local Distance Difference Test). As it name says, this metric is a quality indicator of how good is the model at a local scale. Each amino acid has this score and we can use it to distinguish what regions of the model have been properly modeled and what regions have not. This metric is the score used in the alphafold database to quantify the accuracy of the predictions. As you can see, we have the pLDDT across in the Y axis, while the amino acid sequence is represented in the X axis, with 5 lines representing the 5 models obtained.

Regarding this metric, the developers of alphafold provide the following guidelines:

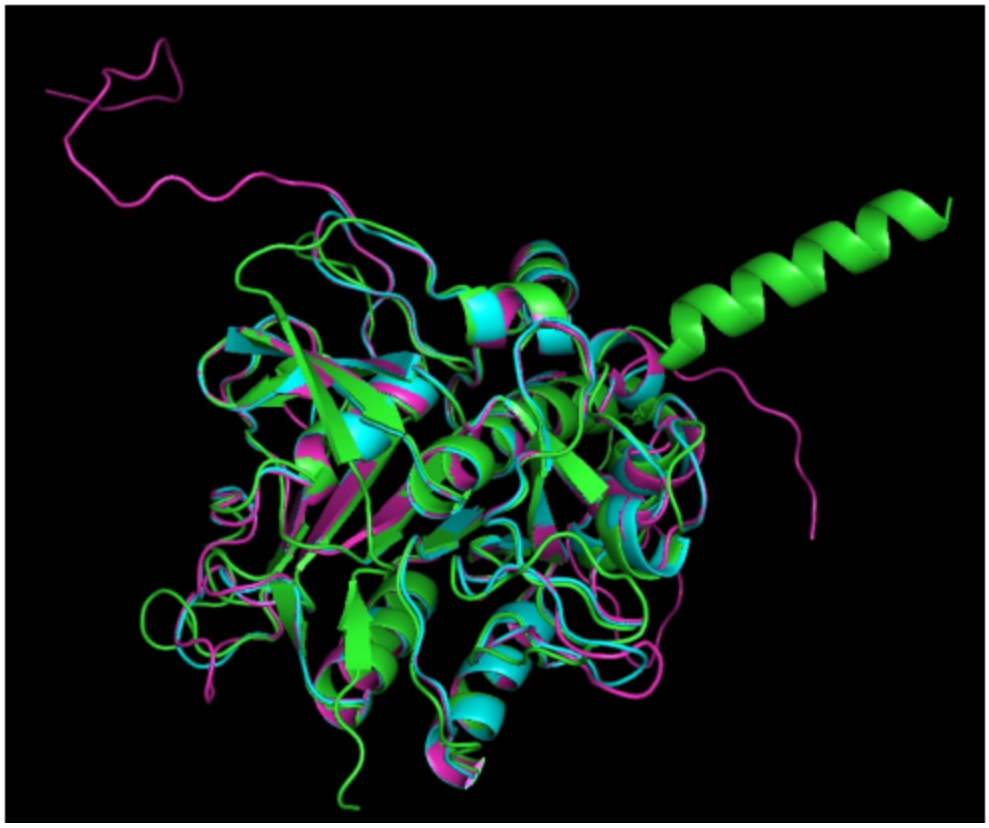
- Regions with pLDDT > 90: Regions modeled with high accuracy.
- Regions with pLDDT between 70 and 90: Regions modeled with moderate accuracy. You can expect a good positioning of the backbone of the protein, but not so much for other features of the protein such as the side chains.
- Regions with pLDDT between 50 and 70: Regions modeled with low confidence, they should be treated with caution.
- Regions with pLDDT < 50: Usually are modeled as loops and they should not be interpreted. However, such low values of LDDT can also be a good predictor of disordered protein regions.

We will start working with some superimpositions:

- Model AlphaFold
- Model we created in practice 4 and 5
- Template

modeller-template RMSD = 0.165; alphafold-template RMSD = 0.564

See that the region that was giving us problems during the modeling now it is different. There is a helix close to that region that is modeled as a longer helix.

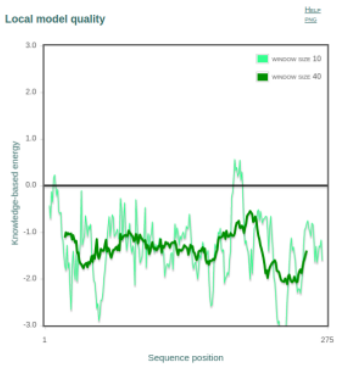


AlphaFold makes the helix longer and makes the loop in normal size. So it is making decisions for us. So now we do not have any crazy loops.

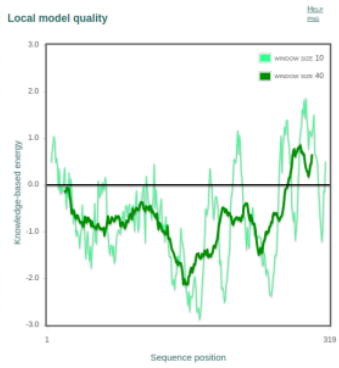
Analyzing the structures with prosa

This is the best way to assess the quality of the models obtained by alphafold and modeller. When doing this analysis, it seems that the PDB files provided by collab fold have a format that prosa web cannot understand. To fix this, you can use the PDB model for this same protein in the alphafold database: https://alphafold.ebi.ac.uk/entry/P11018

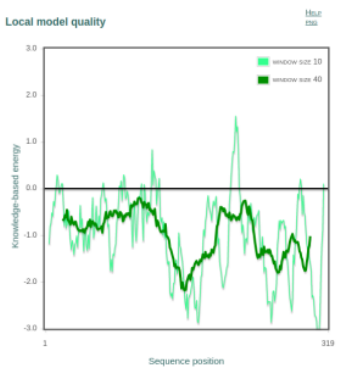
Our template (1meeA)
Z-score = -9.51



Our corrected model (S5)
Z-score = -7.26



Alphafold model
Z-score = -9



So, the AlphaFold model is way better than the one obtained by modeller.

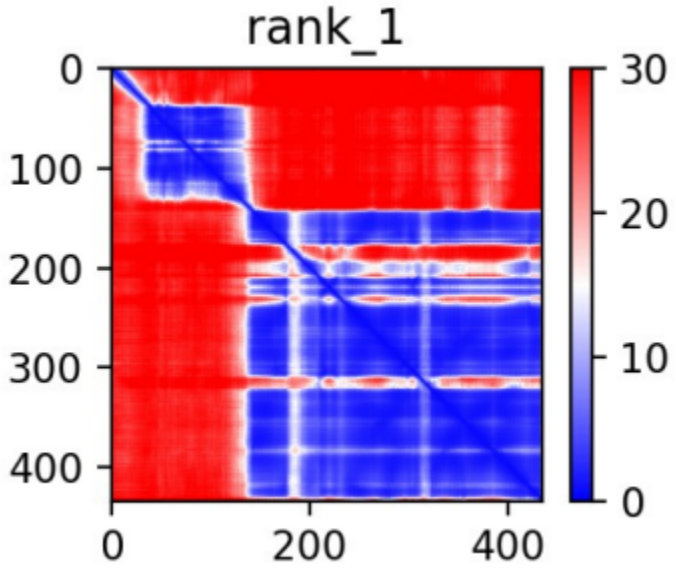
Step 2. Modeling a protein with 2 independent domains

In this step we are going to model a nuclear receptor. Nuclear receptors are transcription factors that also bind hormones. This binding to hormones modulates their activity, it is one of the mechanisms by which our cells can react to some of the hormones we produce, such as estrogen.
Due to this dual function, nuclear receptors must have (at least) two different domains:

- One domain to interact with the DNA (remember that it is a transcription factor)
- And another domain to interact with the hormones.

These two domains are independent from each other and are linked by a flexible region of the protein, which can also be called hinge. The protein we will work with (NR1I2) can be found at the following uniprot page (https://www.uniprot.org/uniprotkb/O75469/entry) and has a DNA binding domain (amino acids 41-102) and a hormone binding domain (amino acids 146-433).

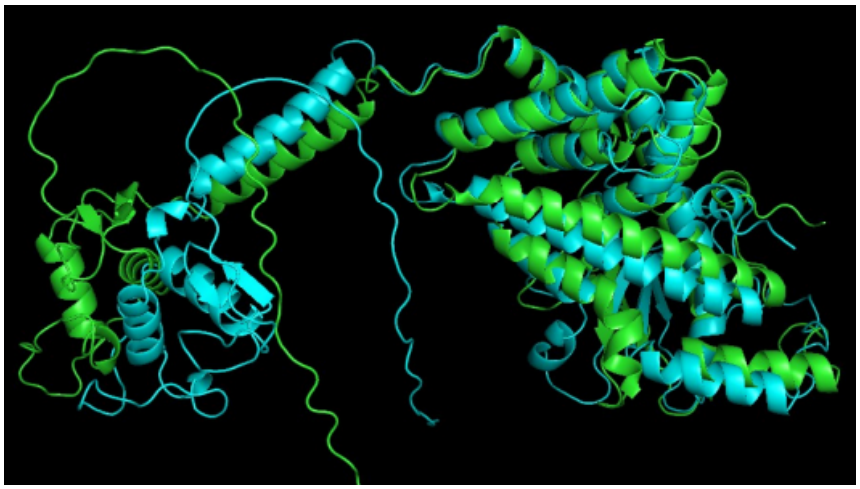
If we check the output PAE plots of alphafold we will see that we can clearly distinguish the two domains and the hinge:



Here we can see how alphafold is very confident of the modeling of the two domains independently. Each one of the blue squares corresponds with each of the two main domains in the protein: the DNA binding domain and the hormone binding domain. The blue regions mean that alphafold is expecting low error regarding the distances between the amino acid positions represented in the X and Y axis. The red regions mean that alphafold is expecting high error regarding the distances between the amino acid positions represented in the X and Y axis.

See that alphafold has a clear idea of where are the two domains independently, but it doesn't know where each domain goes relative to the other. Meaning that the orientation of the domains is not clear.

Interestingly, we obtain a similar situation if we try to superimpose the models that we obtain. Open the two first models in pymol and try to superimpose them, you should get something similar to this:



The superimposition is quite bad, the RMSD is 6.265 Å, and yet the models are very similar if we look at the plots provided by alphafold regarding its predictive performance.

What do you think is happening?
The problem here is the hinge region linking the two domains. This hinge is flexible, and alphafold is quite bad at predicting flexible regions. This makes sense, since flexible regions are hard to crystallize and they are scarce in the PDB (which is the training set used to train alphafold). Also, flexible regions are by definition flexible. Therefore, it doesn't make much sense to try to predict a fixed structure for something that is going to be under continuous change.

See that the angle that the flexible hinge takes is going to define the orientation of one domain towards the other. This makes the superimposition go wrong, but if we superimpose the domains separately, we will see how they superimpose perfectly. You can do so with the following pymol commands:

- To superimpose the DNA binding domain:
  super model1 and resi 41-102, model2 and resi 41-102

See that now the two domains fit way better and the RMSD is smaller (0.130).

- To superimpose the hormone binding domain:
  super model1 and resi 146-433, model2 and resi 146-433

See that now the two domains fit way better and the RMSD is smaller (0.161).
See that when we are superimposing the two domains separately, the RMSD that we get are extremely low, and the domain that is not superimposed is located in very different positions. The idea here is that the domains are properly modeled, but the connection and the orientation of these domains is prone to error.

Step 3. Modeling a chimeric protein (Globular protein + GFP)

Since alphafold only takes as input a sequence, we can put anything we want as input, as long as it is made of amino acids. This means that we can input proteins that don't exist in nature, but that could be created in the lab with the appropriate molecular biology tools.
One of these cases are chimeric proteins.

A chimeric protein is a protein that results from the fusion of two or more proteins. It is a common procedure in molecular biology and usually involves fusing one protein of interest with another protein that we can easily detect. One classic example of a protein that can be easily detected is the green fluorescent protein from Aequorea victoria (also known as GFP). As its name says, it is a protein that can release green light via fluorescence.
We are going to model a chimeric protein made of two parts:

- The serine protease of Bacillus subtilis we modeled in step 1
- The green fluorescent protein from Aequorea victoria.

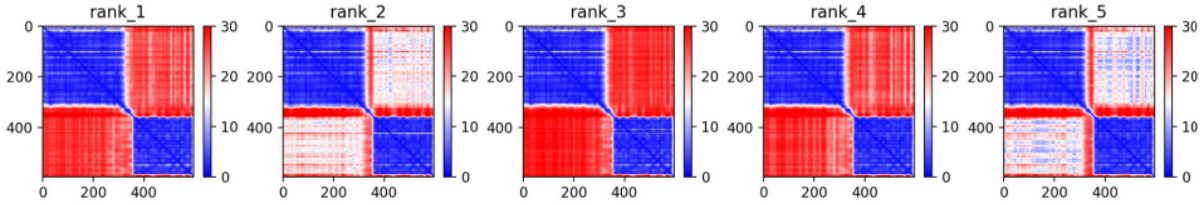
In between the two proteins we will place the hinge region of the nuclear receptor we modeled in the previous step. This will enable that the two domains don't clash with each other and that they have space to operate independently.

Thus, our input sequence for alphafold looks like this:

MNGEIRLIPYVTNEQIMDVNELPEGIKVIKAPEMWAKGVKGKNIKVAVLDTGCDTSHPDLK
NQIIGGKNFTDDDGGKEDAISDYNGHGTHVAGTIAANDSNGGIAGVAPEASLLIVKVLGGE
NGSGQYEWIINGINYAVEQKVDIISMSLGGPSDVPELKEAVKNAVKNGVLVVCAAGNEGD
GDERTEELSYPAAYNEVIAVGSVSVARELSEFSNANKEIDLVAPGENILSTLPNKKYGKLTG
TSMAAPHVSGALALIKSYEEESFQRKLSESEVFAQLIRRTLPLDIAKTLAGNGFLYLTAPDEL
AEKAEQSHLLTLKKEMIMSDEAVEERRALIKRKKSERTGTQPLGVQGLTEMSKGEELFTG
VVPILVELDGDVNGHKFSVSGEGEGDATYGKLTLKFICTTGKLPVPWPTLVTTFSYGVQCF
SRYPDHMKQHDFFKSAMPEGYVQERTIFFKDDGNYKTRAEVKFEGDTLVNRIELKGIDFK
EDGNILGHKLEYNYNSHNVYIMADKQKNGIKVNFKIRHNIEDGSVQLADHYQQNTPIGDG
PVLLPDNHYLSTQSALSKDPNEKRDHMVLLEFVTAAGITHGMDELYK

The amino acids without background color belong to the serine protease, the ones with yellow color belong to the hinge, and the ones with green background belong to GFP.

If we check the PAE plots that we obtain from this modeling, we see a similar pattern to the one in the previous step: the individual models are properly modeled, but the connection between them is not.



We can further prove this idea by superimposing independently each domain of the resulting model with a template:

- For the serine protease region we can superimpose on the chain A of 1mee, the template we used in seminar 4 and 5:

fetch 1mee
super model1 and resi 1-319, 1mee and chain A

See that we obtain a very good RMSD value (0.556).

- For the GFP region we can superimpose on the chain A of the 1gfl PDB structure, which is the structure of a GFP protein:

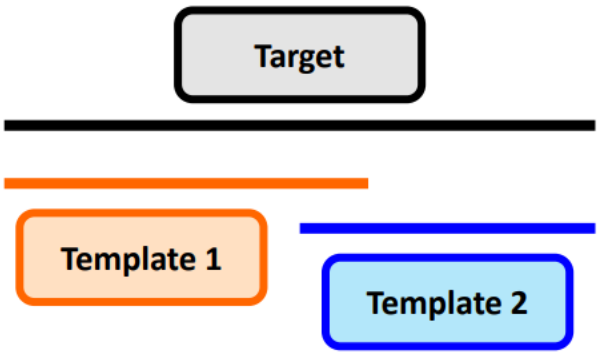
fetch 1gf
super model1 and resi 350-595, 1gfl and chain A

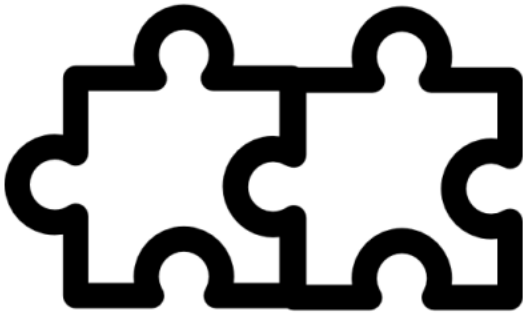
Again, we obtain a very good RMSD value (0.381).

If AlphaFold is so good, why did we use Modeller?
AlphaFold uses homology modeling, but it is not the best way to learn what homology modeling is. Moreover, it has some limitations that can be overcome with Modeller and/or other methods

Modeller performs homology modeling by taking structural information of entire proteins and applying it to targets.

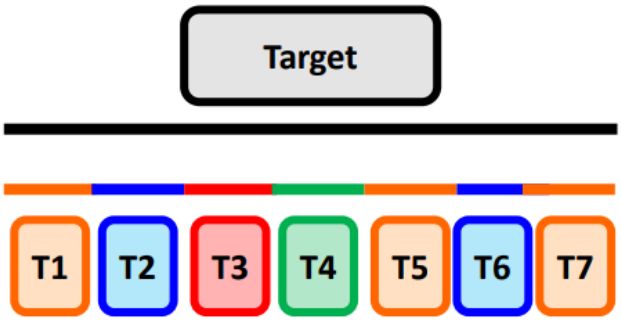
Imagine that we want to model a protein with two domains. The most straightforward approach is to use one template for each domain and join them.

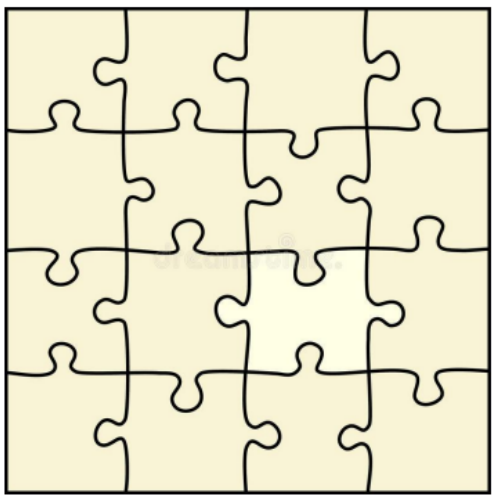




Rosetta performs homology modeling by taking structural information of fragments of 9 amino acids and applying it to targets.

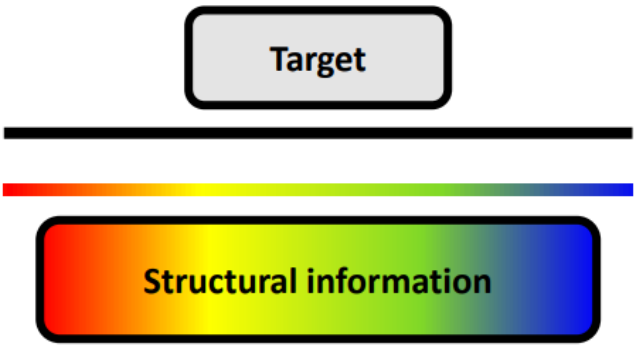
In comparison with modeller, now we are using a larger number of templates and each template corresponds with a smaller region of the target





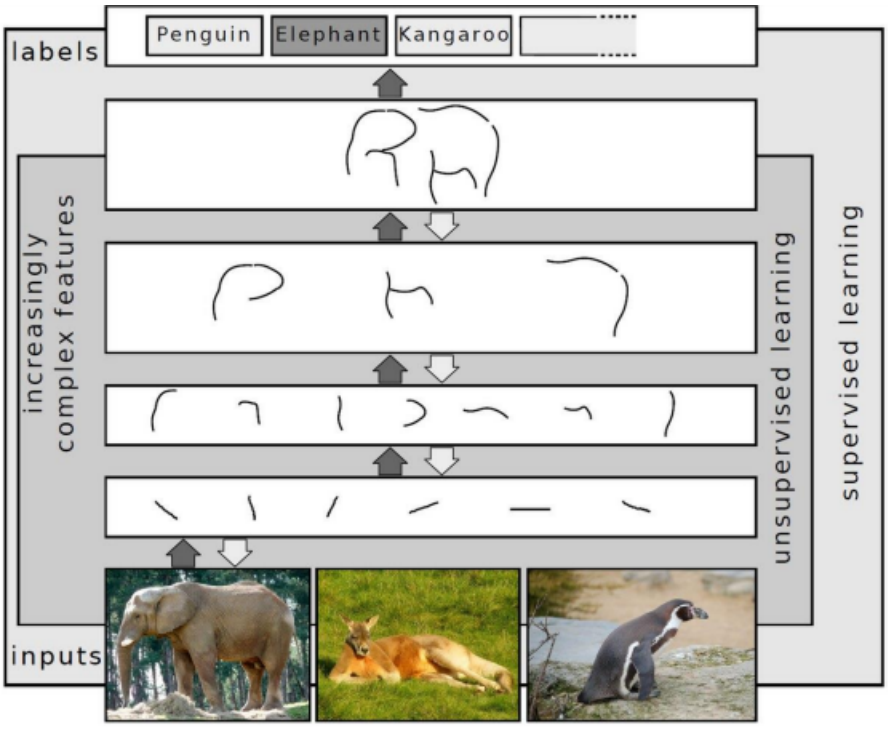
Alphafold performs homology modeling by taking structural information and using neural networks to deconstruct this information at amino acid level.

We are using structural information that is independent for each amino acid, and yet it is wholly integrated to generate a model that makes sense



At this point, alphafold resembles convolutional neural networks for the recognition of images

See how deep learning algorithms are able to extract features for independent elements of the image and then combine them into something meaningful.
Something similar happens with the information per amino acid in alphafold.



Alphafold gives small freedom to the user. This makes it easier to use, but if you don't get the results you want there are few options for improvement.

Modeller gives the user control on what templates to use and gives access to the alignment.

Limitations

Step 4. AlphaFold struggles with proteins for which there is no available experimental structures

Remember that alphafold is a program that requires training to provide predictions. If it has to model a protein that is different from what it was in the training set, it is likely that it will model it wrong. This is the case for the Saccharomyces cerevisiae Sec3 protein, which we are going to model in this section of the tutorial.
Sec3 belongs to a huge protein complex called the exocyst. This protein complex is involved in the transport of vesicles within the cell and it is fundamental for vesicles to fuse with membranes and release their cargo to the extracellular space. There is only one structure of Sec3 in the PDB. It was obtained in 2017 using a technique called Cryo-EM, and in that experiment they obtained the entire structure of the exocyst. Cryo-EM is an excellent experimental technique to determine the structure of big protein complexes, but it has a drawback: the structures generated by this technique have low resolution (although they are getting better). Besides, in the structure of the exocyst, Sec3 is not complete, actually most of the protein is missing.
Now we are going to challenge alphafold to predict a protein for which there is no entire structure in the PDB, and the only partial structure available has low resolution. After obtaining this model we will evaluate its quality by superimposing it to he available Sec3 structure and by making a prosa analysis.

Superimposition: **fetch 5YFP**

Now remove all chains but chain A, which is Sec3. Also, we recommend you to rename the alphafold model to a simpler name, such as model1. Then you can perform a first superimposition with the super command:

super model1, 5YFP

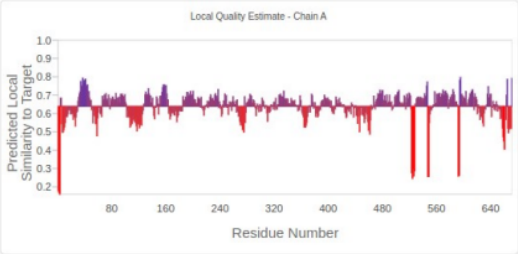
See that this superimposition is quite bad, with an RMSD of 9.860 Angstroms. Since the two proteins that we are superimposing are the same and have identical sequence, we can also use the align command:

align model1, 5YFP

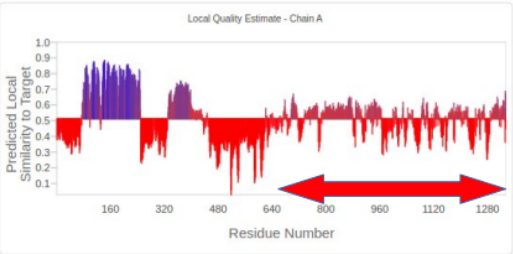
However, the align command provides an even worse superimposition with an RMSD of 10.470 Angstroms. From these superimpositions we can conclude that the alphafold model doesn't resembles the experimental structure.

- Statistical potentials (QMEAN): Prosa-web is not working for the alphafold models, that is why we will make this analysis with QMEAN, another program that uses statistical potentials to score how reliable is a protein structure. See that the statistical potentials values have been reescalated to a 0 to 1 scale, where 1 means good quality and 0 bad quality. The results for the experimental structure can be seen here: https://swissmodel.expasy.org/qmean/tB6Apb.
  The results for the alphafold model can be seen here: https://swissmodel.expasy.org/qmean/LUZjUA.

Experimental structure (5yfp, chain A)
Global quality: 0.64/1.0



Alphafold model
Global quality: 0.51/1.0

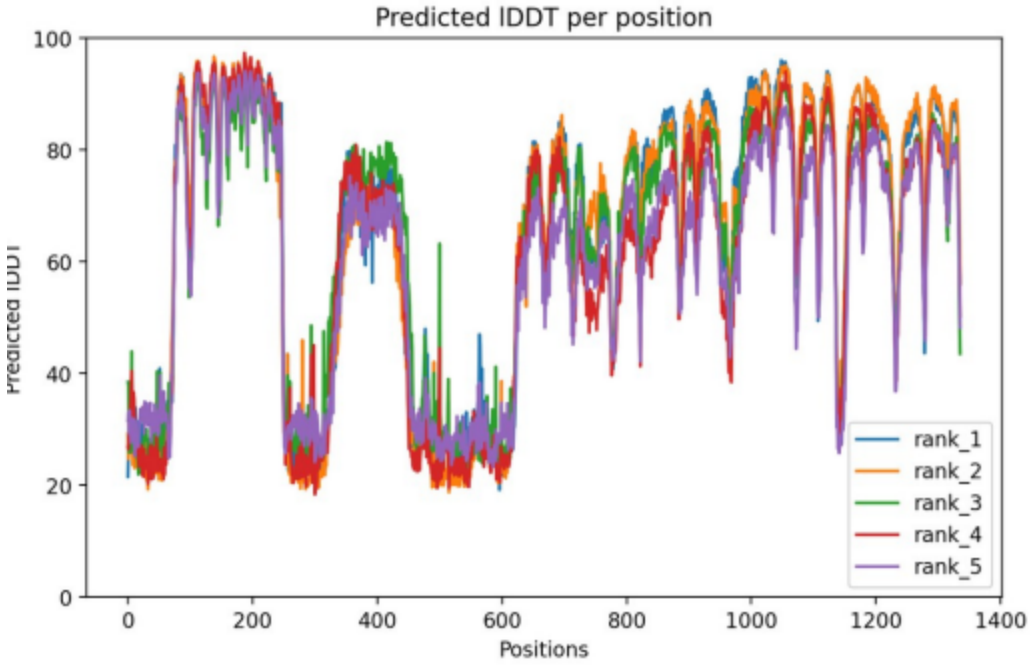


The experimental structure is a 6.4 out of 10 (statistical potential) because, as we remember, statistical potentials take 2 things into consideration to score protein structures:

- Distances between aa
- Exposure of aa to water

So, we are taking into consideration that some regions are exposed into the solvent when in reality they are not, because they are forming the complex.

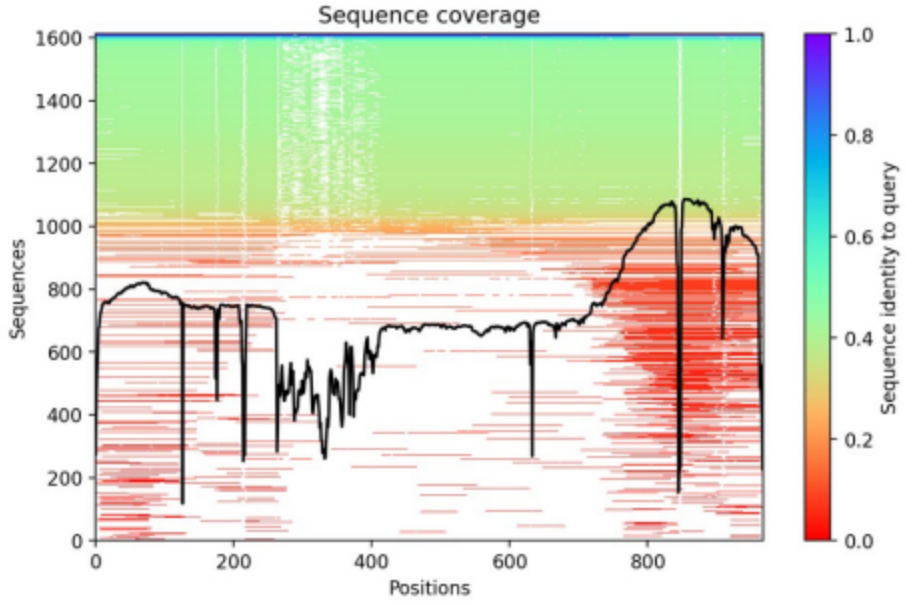
Here the superimposition and the analysis with statistical potentials have consensus in their results: both of them indicate that the alphafold models are not correct. We can also get this idea by checking the LDDT plot, that looks quite bad. If we pay attention, we will see that there is a correlation between the QMEAN score and the pLDDT provided by alphafold:



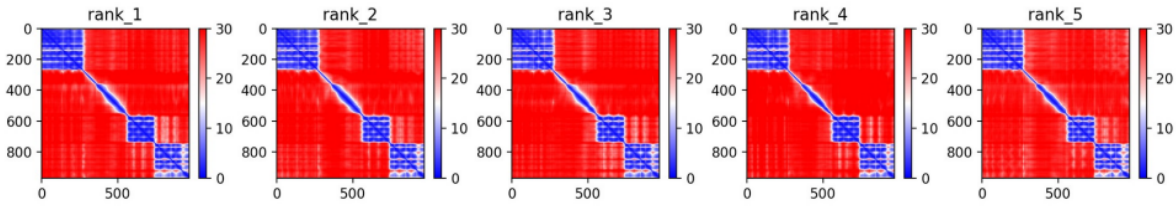
Step 5. It does not take into account biological characteristics of proteins. Handles membranes poorly.

Aspects such as the function of the protein or where is this protein located within the cell are not considered to make this modeling. This becomes a clear problem when modeling transmembrane proteins. In this step we are going to model the protein Sla2 from Saccharomyces cerevisiae. This is a huge protein with a transmembrane domain in the N-terminal region. If we look at the plots provided by alphafold, we will get hints that the modeling didn't go as well as we would like:

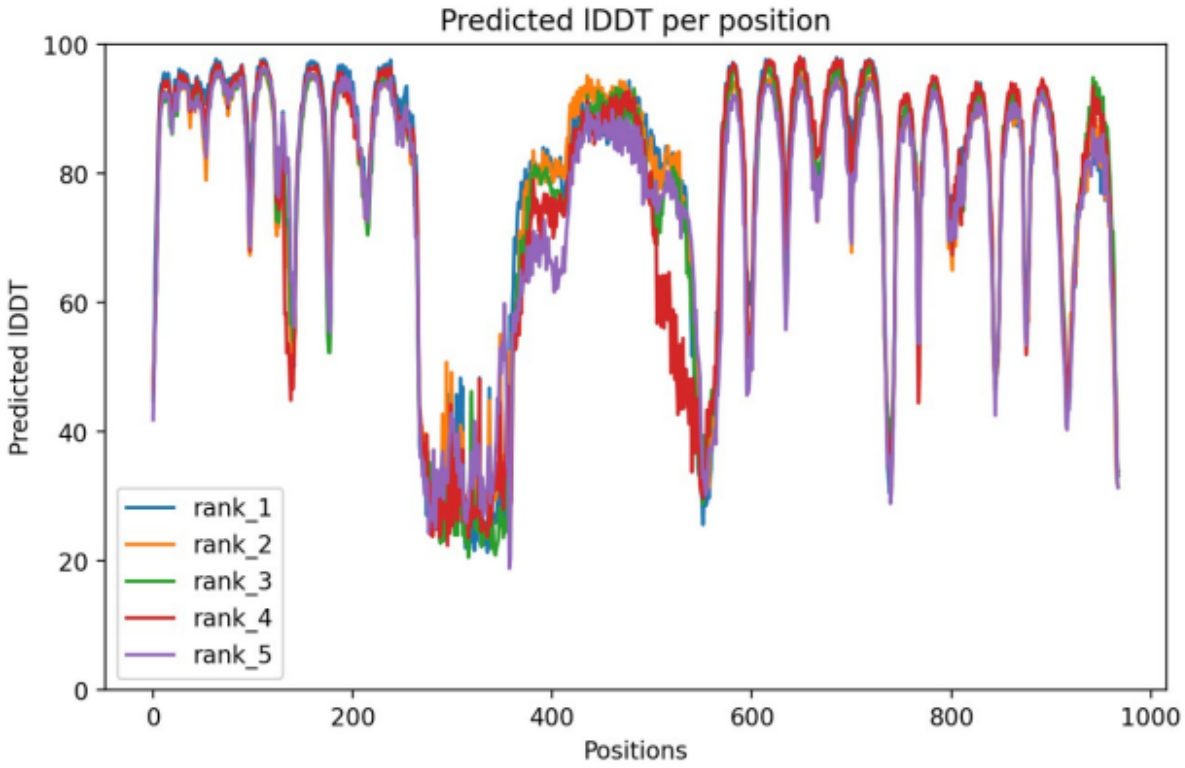
- Coverage plot: We see that the sequence coverage for most of the protein is not as high as it was for the other proteins we have analyzed.



- Predicted align error plots: Most of the plots are in red, with small blue regions indicating folds that have been properly modeled. There are regions of the protein where we only see a blue diagonal, this means that the modeling in this region is not reliable.

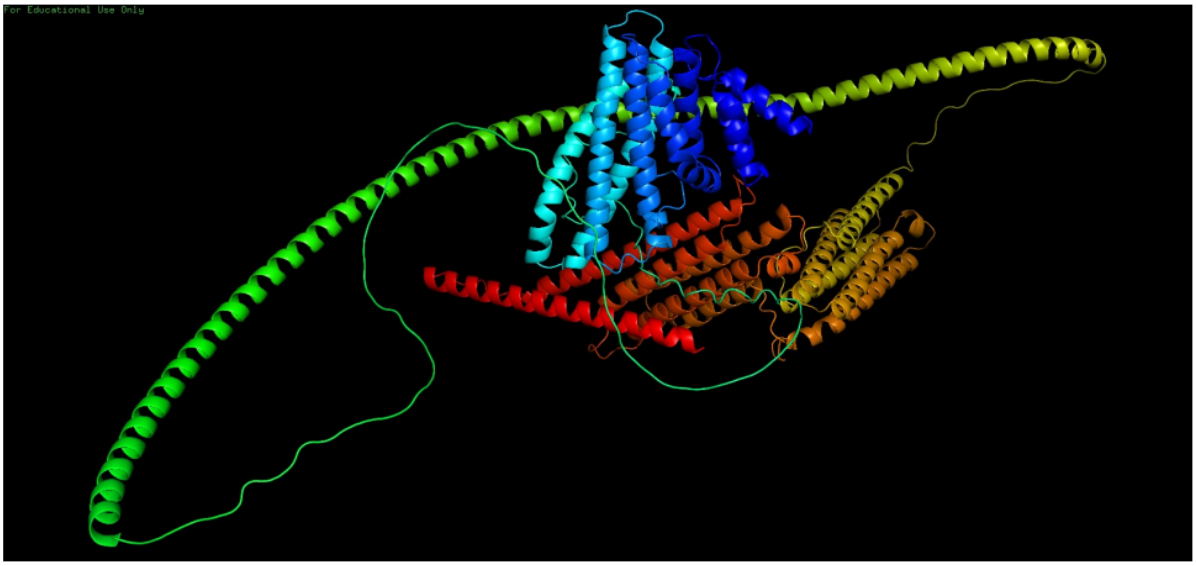


- Predicted LDDT per position: Some regions have high values of LDDT, but also we find regions with extremely low LDDT. Regions with low LDDT are not reliable (or are intrinsically disordered). Usually these regions with low LDDT are loops, and if they are not properly modeled they are going to affect the whole model. This happens because loops can turn, while helices and sheets can't. Therefore, the loops determine the orientation of sheets and helices within the structure. If the orientation of a loop is not correct, the helices or sheets that are connected to it will not be correct either.



Also, if we know that in the N-terminal there is a transmembrane domain, we can visualize the protein with pymol and try to identify at what position the membrane would be. In the following image you can see the model of Sla2 with rainbow coloring going from blue in the N-terminal to red in the C-terminal. In the center of the image we see the N-terminal in blue with several helices that could constitute the transmembrane domain.

If this blue region is the transmembrane domain, do you think is there anything wrong with this model?



As you can see, if this model is alright it means that the huge helix we see is inside the membrane, parallel to the surface, and this doesn't happen in nature. The point here is that alphafold is not assuming that there is a membrane somewhere in the structure, and since it is missing this information, it is allowing the model to roam free in regions where a membrane should be placed.
This is not a dramatic error, since loops are flexible, we could obtain a more suitable conformation using a molecular dynamics simulation.
On the other hand, the long loops in this model that correlate with low pLDDT values suggest that this model is not fully correct, and that these long loops should not be trusted.
We can see a long loop from amino acid 270 to amino acid 357 that correlates with very low values of pLDDT. Also, the long helix that comes after (from amino acid 357 to amino acid 546) has lower pLDDT values than most of the domains that we have predicted correctly during this tutorial.
This point suggests that for transmembrane proteins, alphafold is not the best option and experimental methods such as X-ray crystallography can still provide structures that wouldn't be available without it. Also, if templates are available, traditional homology modeling methods such as modeller are still useful.

Step 6. It is not good making models of disordered proteins (although no program is)

Intrinsically disordered proteins (or protein regions) don't have secondary structure. They are just long loops which are very flexible and variable. In some cases, these proteins can adopt a specific protein folding, but they require the interaction with other proteins for this to happen. Intrinsically disordered proteins are hard to study. Since they are in continuous motion, they cannot be crystallized, and the main experimental method to characterize them is nuclear magnetic resonance (NMR). Also, when we try to model them, most of the times we get long loops (which is the same that you get with modeller of with alphafold when the modeling goes wrong).

Now we are going to model the thylakoid soluble phosphoprotein (TSP) from Spinacia oleracea (Spinach). This is an intrinsically disordered protein for which we have an available structure in the PDB obtained by nuclear magnetic resonance (https://www.rcsb.org/structure/2FFT).

NMR structures provide an ensemble of structures. We can see that this ensemble is very diverse in the case of TSP, as its loops have a lot of freedom to move. We see something similar in the models obtained with alphafold. There is only one region that doesn't change: a small helix located at the center of the protein.
We are going to analyze this experimental structure with pymol and then compare it to the models of alphafold:

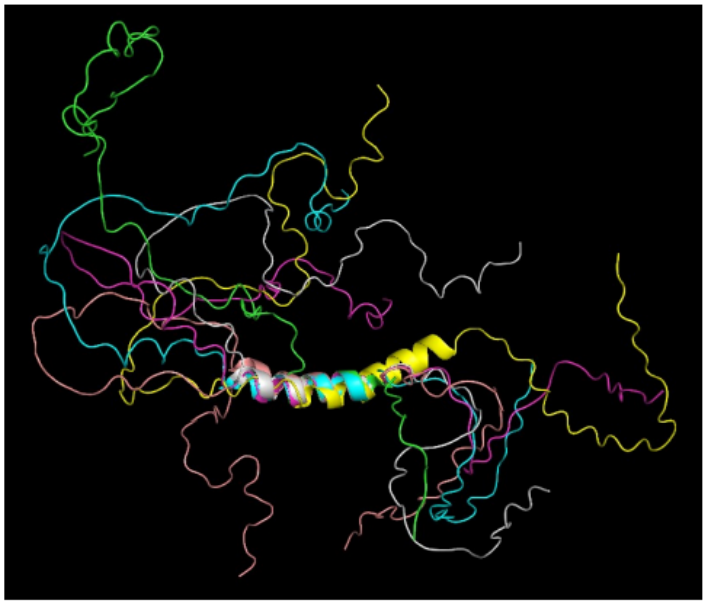
fetch 2fft

You can see the different conformations detected by NMR bi clicking into the play button you have in the low right corner of your screen. In total, 20 conformations have been obtained for this structure.

Now, open the alphafold first model and rename it to model1. Then superimpose it to the experimental structure:

super model1, 2fft

As you can see, the only regions that are superimposing properly are the helices at the center of the protein. If you load the 4 other models generated by alphafold you will see how their central helices superimpose perfectly in the experimental structure, while their loops adopt random conformations. Load the 4 models left and superimpose them to the experimental structure, you should get something like this:



Interestingly, alphafold is predicting this structure correctly, because it is a disordered protein. The problem is that it is hard to know when alphafold is predicting a disordered protein correctly and when it is doing so because the model it makes is wrong. In both situations the confidence values are very low. That is why, the use of NMR and other experimental structural methods can be useful in many scenarios despite the huge success of alphafold.

Conformational space

Why simulation is important?

Simulation is a little bit different from what we have been doing so far. We have learned to predict structures, modeller… so we have learned different ways to get structures.

Here we are generating ENSEMBLES using simulations. We take a PDB and create an Ensemble, which is the best representation of a molecule in solution. So, we are getting close to the behavior of the protein in solution.

So, when we talk about simulation, we are not just talking about operations to make new proteins, operations to predict the structure… but we are talking about a kind of "computational microscope". It's a real experiment done in silico, which starts to have similar scopes of the real experiment that we want to do in the lab.
MD Simulation is mature enough to give insight in biological problems

- "Computational microscope"

Note that if we think in simulations as possible experiments, we do not have to buy anything (just computers). So, we have more freedom than a wet lab and therefore we can explore more solutions.

Simulation experiment can be designed to analyze situations not accessible in "wet" experiments

- Non-existing ligands
- Explore conformational freedom
- …

Simulation experiments help to define situations/states that can be tested in the lab.

Which algorithm?

There are 3 ways to make simulations:

- Molecular Mechanics
- Molecular Dynamics used to make the big simulations
- Monte Carlo

Molecular mechanics: It is always the first step because we need to find a mathematical minimum (we need numerical stability).

- To find the most stable structure. Optimization of the energy function.

Monte Carlo: We do not use it to get a representative ensemble of the states but to switch among different possible trajectories. So, it's a tool to help MD to do the right simulation. Meaning that it is used just to expand the possibilities of normal MD.

Molecular dynamics:

- Behavior of the systems along time.
- Folding/unfolding
- Optimization
- Principal components and Flexibility analysis

Conformational ensembles

"Ensemble": set of structures that represents ALL possible microscopic states of the system

Thermodynamics can be deduced from the average of "ensemble" properties using Boltzmann. Remember that we are always jumping from the microscopic to the macroscopic (thermodynamics) states.
We measure in the lab microscopic properties and we simulate microscopic systems and we play with this dual behavior.

Ensembles help to understand macromolecules behavior, since it is the best representation. Remember that we were talking about conformational selection → When a protein binds a small ligand, it is selecting one of the possible states because that state is the one that has the better affinity. Because we are selecting a single state, the delta G of that state goes down and eventually it becomes more populated and all the protein is binding the ligand.

- Induced fit vs conformational selection

If we have the option of analyzing these states beforehand, we may evaluate the difference in energy and see which one is the correct.

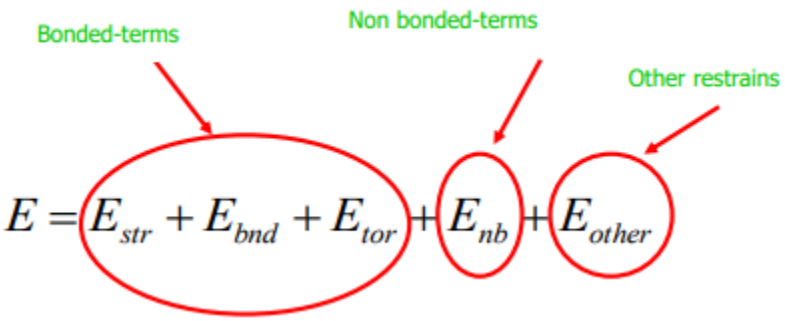
Types of ensembles that represent reality differently:

- Microcanonical ensemble (NVE)
- Canonical ensemble (NVT)
- Isothermal-isobaric ensemble (NPT) → Best represents reality (lab conditions)
- Isoenthalpic-isobaric ensemble (NPH)
- Grand canonical ensemble (μVT)

Classical mechanics (Force Fields)

Empirical approximations (requires parameterization)
Computationally very efficient
No size limit
Less accurate than quantum mechanics
Solvation and entropy cannot be included!!

A force field is a huge formula that allows you to calculate potential energy starting with the geometry of the molecule (PDB).

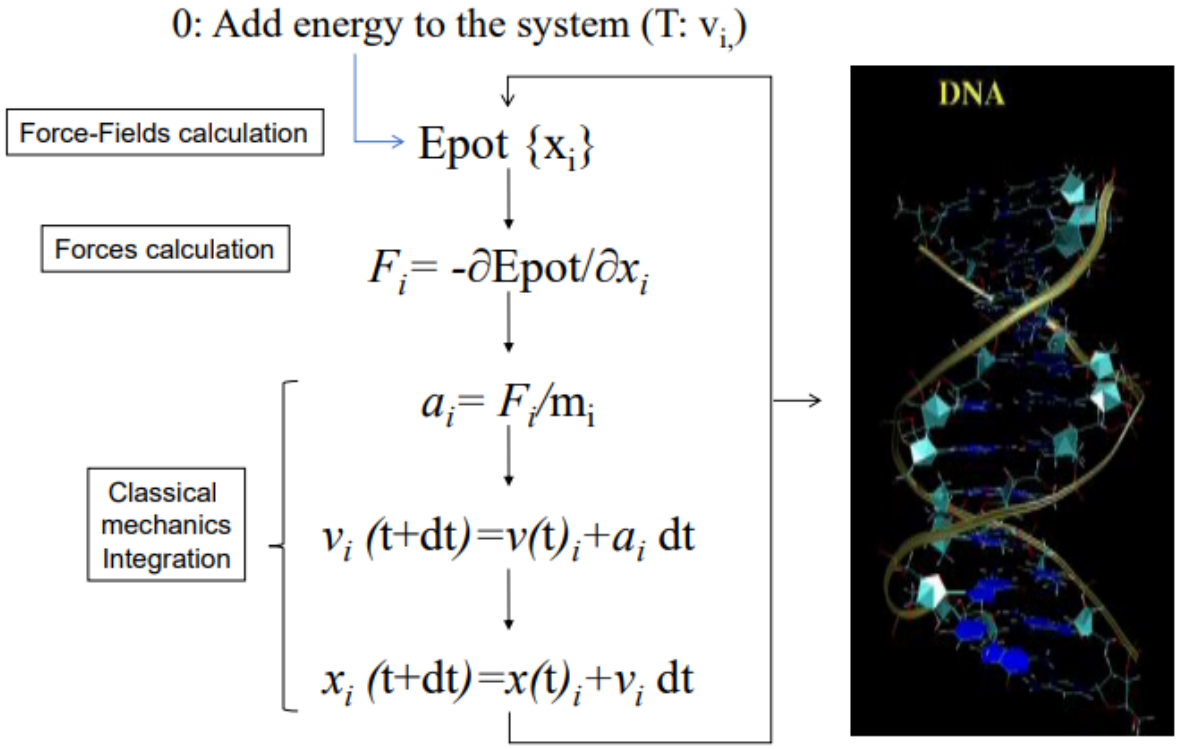


Molecular Dynamics Algorithm

The first step of the MD algorithm is to add energy to the system, which is added in the form of velocity (we allow the atoms to move). This is due to the fact that there is direct relationship between the velocity of the atoms and the temperature. So, we can simulate any temperature by giving the atoms certain velocities.

With this, we calculate the Force Field, the forces (gradients, accelerations…) and we iterate this millions of times to finally get a "movie". We obtain a series of positions of the atoms that change over time. This is what we call "trajectory".
The different conformations in the trajectory are called "snapshots".

Each iteration lasts 1 femtosecond (10^-15 s).
A normal simulation lasts 100 nanoseconds (1e^-9)
So, it takes a lot of simulations.



As we have seen, mathematically speaking, the algorithm is very easy. But we will later face a number of issues.

Experiment planning

1. Define the problem to solve: Translate the biological problem to the simulation world. Find if a mutation is causing a disease, for example.

2. Define the experiment:
    - WT structures vs mutants. If I take the protein and make the mutation, will this change modify the stability of our protein (doesn't fold → doesn't work), which will lead to the disease?
      The simplest thing is to take the protein, make the mutation and go to PyMol. We check if there is any problem (new aa is to big, for example).
      Check if the protein will change its shape so that the new aa fits, using simulations. Experiment:
        - Put our WT in water
        - Put the mutant in water
        - We allow them to relax. If both molecules are stable we can just compute the difference in stability.
    - Apo structures vs Holo, alternate ligands
    - Alternate conformations
    - …

So, I need to have clear what I want to do. That's a list of simulations:

- Which is the system I want to simulate and which method I will use.

3. Choose the structure(s):
    If we are trying to understand the binding of a ligand, we will make a simulation using the protein with and without the ligand.
    If we try to understand a complex, then we use both monomers separated and both together…
    - I have to choose which structure I will use → Go to the PDB, check the resolution, quality, the presence of ligands if needed…
    - Also I need to decide which is representation level, which in general will be atomistic unless my protein is too big and my computer can not work with all the atoms. Then I would use coarse-grained.

4. Choose the simulation strategy: Plain simulations (the system can move freely), biased (do I need to trick the simulation a little bit because I am trying to see a conformational change that will never happen unless I push it), coarse-grained.

5. Choose the analyses to be done:
    - Geometric measurements: In the case of WT vs Mutant → Check if the mutated residue is stable during the simulation or if it is changing the shape (not stable). This can be done measuring distances, angles…
    - Clustering: Generating different states (will see next day)
    - Energetic analysis
    - Flexibility and movement modes

In our example, the key point is the stability of the 2 proteins.
If I am calculating stability, do I need to do simulations of the unfolded state? Remember that stability is always relative to the unfolded state. So, yes.

Steps

Once I have decided what I want to do, I need to:

1. System setup: Required to have a stable system to start the simulation

2. Simulation (done by the computer, you just click a button):
    - Usually the longest part of the work (mostly unattended)
    - Simulation replicas recommended. Simulations are stochastic (no simulations are repeated), meaning that if you run 5 simulations in the same conditions you will have 5 different results (replicas).
    - Plan computational needs in advance
        - Limits on queuing systems, limits on allocated resources…

3. Analysis: Once you have done the simulation, you have to analyze it.
    - QC: Check that the simulation is correct. Reproduce exactly what is in the PDB (because that is the real structure). If our simulation is not able to replicate the experimental structure you are in trouble (something is wrong).
    - Extracting information from the trajectories using the list of analysis that you may have (using IA nowadays).

4. Redesign, repeat if the result was bad until you get the correct one.

System setup

It is the first part, fully mandatory: Aims to prepare a system as realistic as possible.
We start with a PDB and we end with a protein in a box of water, ions…

This is the system that we are putting in the computer to make the simulation.

Why is required?

Experimental structures can have problems, they are not perfect. Because:

- The experimental guys are not perfect
- Also the X-ray (can't see H for example so we need to add them manually), NMR (if the proteins move a lot, then the signal is bad)… has limitations.
- There are also mutated residues, incomplete residues, missing residues that you need to complete.
- You are not always sure of which is the real oligomer state (you have the data and some predictions but you do not know if it is a dimer, tetromer… in solution because you just have the crystal with the structure), pH/pKa (since we do not have H, we do not now if the histidines are protonated, for example), disulphides, covalent linkages ions/solutes.

Remediated PDB (since 2007) → They corrected the structures in terms of:

- Atom labels, sequences / numbering, ligands, …

As mentioned, experimental structures may be:

- Incomplete (hydrogens are never there, mobile fragments missing, side chains missing)
- Wrong (Vdw clashes, atom labels, geometry)
- All X-ray structures are over compacted because the X-ray limits are less estringen (they don't calculate VdW and they crash).

Xray structures are structures "in a crystal", not in solution

- Wrong contacts, aggregated states, crystallization conditions, pH
- The pH is normally wrong in NMR because to make the protons more stable, NMR is done in an acidic pH.

NMR structures may have low coverage in flexible regions

So, our simulation will require:

- A complete (and realistic) structure (all atoms should be present)
- "Structure in solution" (+ ions, water, …)
- Ligands if necessary. Problem: Since they are not amino acids or nucleotides, so they may not be parametrized in the force fields. So, if you have a new molecule, you will need to find the parameters for the force field of that molecule.
- We need to start the simulation with an energy minimum according to the forcefield to avoid numerical instability. Make the derivative of the potential versus the coordinates.

Initial steps of simulations should include

- **Minimization** (Molecular Mechanics) to remove energy hot spots
- **Equilibration to relax** structure and allow new atoms to accommodate
- Equilibration should **keep the experimental** coordinates

Completing the structure

So, the first thing is to complete the structure.

Checking structure quality:

- Resolution, completeness
- Validation reports at PDB
- Automated checkers (biobb, MDWeb)
- Possible issues: Alternative conformations, clashes, wrong atom labels

Missing residues, side chains:

- Disulfide bonds should be built
- Side-chains can be reconstructed automatically
- Missing main-chain should be built (usually by comparative modelling)
- Incomplete terminals could be capped to avoid additional charges

Hydrogen atoms:

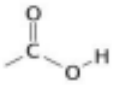
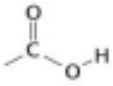
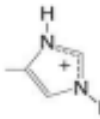
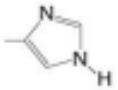
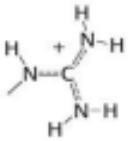
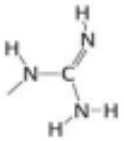
- Most hydrogen atoms can be added automatically.
- Ionic residues should consider pK values and experiment pH
- Environment changes pH
- Watch for tautomeric forms
- **Ionization states cannot be changed during simulation (if you protonate a histidine, it will be protonated in all simulations)**

Setup: Protonation state

We are adding H and we need to protonate or not some residues.

If we are at a pH of 3, the aspartic acid is going to be protonated or not (its pKa is 4.1)? We are going to have the protonated form because the solution is even more acidic than pH=4.1.

TABLE 3.1 Typical pK_a values of ionizable groups in proteins

Group	Acid	⇌	Base	Typical pK_a*
Terminal α-carboxyl group		⇌		3.1
Aspartic acid Glutamic acid		⇌		4.1
Histidine		⇌		6.0
Terminal α-amino group		⇌		8.0
Cysteine		⇌		8.3
Tyrosine		⇌		10.9
Lysine		⇌		10.8
Arginine		⇌		12.5

pK_a depends on T[a], ionic strenght, and environment

Obviously, lysine and arginine will always be protonated.
pKa depends a little bit on temperature, a little bit on ionic strength and a lot in the environment. Everything that is around its side chain will change the pKa.

Setup: Water and ions

Until now, we have completed the structure, added H, protonated the correct residues… it is important to look at the PDB record and check if the experimental pH is there. Now we must add the protein to our solution.

Structure in solution requires a solvent environment → We need to add water and ions.
We add ions because we have salt in our organism (0.15M, which is high). The thing is that proteins are not very sensitive to ions, but nucleic acids are (because they are - charged and they need + ions to compensate).
If you go to the structures, they also have water. They are called structural or crystallographic water molecules and, if they appear in the crystal, it means that they have an electron density that is feasible in the crystal. The crystal is full of water, but only the water molecules that are fixed in a specific position in all the copies of the crystal, accumulate enough density to be seen. So, if we see a water molecule, it means that it is highly important because they are bridging H bonds between 2 parts of the protein (for example).

- Must be kept when relevant for the structure.

On the other hand, many of the water molecules found in PDB are wrong (they are invented). The reason is that it is a very good way to hide effects on the X-ray. If you have problems with electron density (you have noise), you can put water in and the noise will disappear. For this reason, some people remove all HOH. Then you can put the waters back in the best possible positions:

- If we add a water molecule in X position, will it make a good interaction with the environment or not? If it is good, then we add the water.
  We know if the interaction is going to be good or bad by simply calculating energies and force fields.

ation

- Water and ions added in order covering the most energetically favorable positions

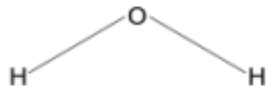
When everything is right, we add bulk water:

- Protein is soaked in a box of equilibrated water molecules. Colliding Waters are deleted.

Water models

Water is the most abundant component in simulations, so we will spend a lot of time computing things on water. So, let's use waters that are easy to calculate → Water Models

We are not interested in the water itself, so they will act as rigid bodies. This will make the simulation faster, and help to improve the water effects.

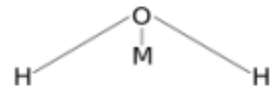


3-site

Most of the force fields have special potentials for Wat-Wat interactions, which are not relevant.

3 site models (3 atoms) are the most typical:

- Rigid, meaning that the distance between the atoms is fixed. It will act like a triangle.
- Partial charge on each atom, VDW only in O-



4-site

4 site models:

- Improve the electrostatics by adding a 4th pseudoatom

Solvent strategies

How many water molecules do we need to add?
It depends. We have many strategies.

Let's say that we do not like water because we are working with a small computer. Then we will have **no solvent**, which is not realistic:

- But it is valid in gas-phase simulations (Mass Spectrometry where water is vaporized)→ But only in this type of experiments where there is no water it is a good approach to not have a solvent.

Implicit solvent (Generalized Born theory): We know the properties of the water, so we may simulate its effect with some numbers. Generalized Born formula is used to calculate electrostatics energy.

- Faster because there is no water, but lacks water entropy. So, we are reproducing the electrostatic effects of water but we have no water.
- Instant (too fast) response
- This is used in experiments where the entropy of the water is not important. For example, simulating a conformational shift.

Explicit solvent: This is the most realistic simulation. Meaning that we are adding water molecules.

- It allows the recovery of solvent entropy and hydrophobic effects.
- Cap → Sphere of water on top of the active site. This was used when computers were small. They are no longer used because they require a lot of work to stabilize the region that does not have water and we have computers powerful enough.
- PBC → Periodic Boundary Conditions

Ions are added substituting a water molecule:

- Na+ and Cl- added to neutralize the system (total charge of the system should be 0) or to achieve a given ionic strength.

How many ions do we add? At least to neutralize the system, otherwise we will have problems. So, if our protein is -8, we will need to add 8 Na+

Also, we may decide to mimic the real physiological ionic strength (0.15 M), so we will need to add more Na+.

Periodic boundary conditions

It's a mathematical trick to add solvent in the simulation.

You have a system that has some specific shape (a cube, for example) and we replicate in all directions the water that is in that cube.

Imagine that the protein moves during our simulation. No problem because the cell in the right will also have the water. So it is a way to mimic “infinite” solvent.

This introduces the concepts of volume and pressure, because: Since each cube has a fixed volume, we can compress or expand. So, we can simulate pressure.

System is replicated mathematically in all space directions

- Atoms can interact with other real atoms or with their “images”
- Atoms leaving the host cell, reappear
- One side effect of this is that we have periodic conditions, thus we use periodicity-based algorithms (Fast Fourier transform) for long range electrostatics.

Usual shapes

- Cubic
- Truncated octahedron

Computational implications

- At the beginning you do not know where your protein is in the cube, so you have to center it.
- Multi-chain molecules (esp. DNA) may split in separate images

Temperature, Volume and pressure

Now we have water, we have atoms… and now we must decide which is the ensemble we are going to simulate → NPT

The initial step is to add random velocity to atoms to match a given temperature and we do this with the Maxwell-Boltzmann distribution.

Pressure is changed by changing Volume

Simulation equilibration

At the end, we must equilibrate the system to start doing simulations again.

- Necessary to avoid artifactual changes due to the setup
- Especially important for atoms added during setup (H, Wat, missing side, mutated sides, etc)

So, this are the things you need to do to equilibrate the system:

- Minimization using molecular mechanics (the derivative of the energy against all the coordinates should be as close to 0 as possible). This allows us to remove any VdW collision that may happen, but without changing too much the coordinates of the structure.
  Moreover, this is really important to avoid problems in the second iteration.

- Increase the temperature gradually maintaining experimental coordinates restrained. We add it slowly because we do not want to disturb the structure too much (we want to keep the original coordinates).
- Decrease restrains gradually
- Free simulation until systems are stable

Guiding MD

Helping the simulation to cover a biological process.

When we do a simulation, the protein does not change its conformation too much. For example, let's say that we want to simulate how a protein interacts with its ligand. To do this, we need to open the active site and therefore change the conformation of the protein.
But as I said, in a normal simulation this will not happen.
So, we need to guide the simulation in order to change the conformation fast.

We will now see a number of tricks to make this happen.

Constrains, restrains

The first trick is to define constraints and restraints.

Constrain: Fixed degree of freedom

- Used to fix coordinates that have unrealistic environments (the cap, for example, we fix the structure that is outside the water) or to avoid unnecessary calculations.
    - The most important one is to keep known geometries: If you are interested in a big conformational change, I will not be interested in the migration of 1 side chain.
    - Bond lengths/angles fluctuate with high frequency without affecting conformations.
    - Algorithms to keep bond geometry allow to increase the time step
    - So, we just fix the geometry because we do not care about the rotations, bond lengths, angles… they are not relevant for the simulation

Restrain: Degree of freedom that is kept nearly constant using potentials.
We use potentials to guide changes. For example, if I want to add external information because I know that 2 atoms are at a certain distance, I will add a potential that will allow the force field to optimize that distance.
If I am at the right distance I do not add any energy, if I am too close I add a repulsion energy and if I am too far I add an attraction energy.
So, we are adding terms to the force fields to add restraints that will guide the simulation.

- Geometrical or “ensemble” restrains
    - Potentials used to keep positions, angles, move to a reference geometry, ensemble properties, etc.
    - Allow to add external information (refinement Xray, NMR), force conformational changes, keep known interactions, …

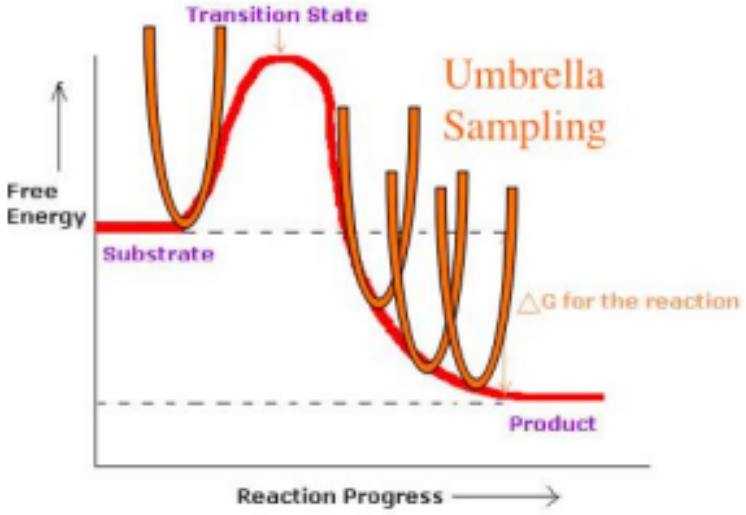
Biased MD simulations

This is a more complex way to do the same thing.
Aim: to favor a given evolution (transition) of the system

Umbrella Sampling

- System is forced to move following a reaction coordinate
- Energy required conforms a Free Energy profile of the transition

I have an initial conformation and a final conformation. There is a transition state in the middle (the conformational change needs a big jump of energy).
The probability of going to that transition state is very low (energy is very high) in our simulation.
We will add some energy.



Targeted MD (most used)

- Potential restraints are used to reduce distance (RMSd) to a given target

Steered MD (most used)

- Forces are introduced to move in a reaction coordinate

Maxwell-Demon MD

- Select the appropriate snapshots that approach the right direction
- If I go to the right direction I continue.

Computational improvements

We will see tricks to make things faster.

Computational challenges

- Simple calculations but large N
  - Bonded calculations O(N) → FINE
  - Non-bonded calculations O(N^2) → Very bad
    These are the VdW and electrostatic interactions, which consume 90% of the time.
    - So, the first trick is to not calculate the NB interactions. If something is to far away, we do not calculate the VdW interaction:
      So, we maintain a list of NB pairs within a distance cutoff (VdW at 10A is 0) and these are the only ones we calculate VdW interactions.

- In Bioinformatics, we always calculate the distances between atoms.
  - But this implies Sqrt (need to check if the number is negative… so it takes even more time. In our case, distances are always positive, so we do not need to do this) and 1/x !!! → Terrible operations
  - Add-hoc math libraries allow to increase speed. For example, it uses a sqrt that does not check if the number is negative.

- Level of floating-point precision
  - Errors may accumulate after millions of iterations

Optimization tricks: Parallelization

It consists in separating a specific work into subprocesses that run in different computers.
There are 2 main ways of parallelization.

How can we distribute the processes?
If the cores are in the same physical machine, they may share the memory.
But memory can not be shared across different computers.

So, in our laptops we have 4 cores so we are parallelizing up to 4 times.
This option is called “OpenMP”, that means shared memory (they share the data through the memory where it is located). This is the simplest way of parallelization.

There is another way that is much more popular, which is called “MPI” (Message Pass Interface).
The problem of having several processors doing something is that they need to communicate with each other. Because I need to put in common the things that different cores are doing.
If everything is in the same memory, then this does not happen.

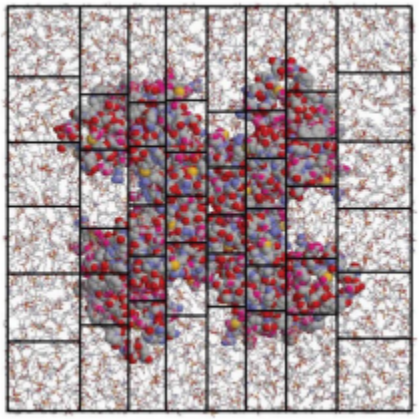
- Heavily limited by communication bandwidth

The most natural way of parallelizing MD is to separate the atoms, so that each core works with X number of atoms.
But you do not know which atoms are going to be still and which atoms are going to move a lot. Meaning that the balance of work for each core is going to be different.

Another problem is that it requires an "all-to-all" update of coordinates. If I change an atom so that it interacts with the rest of the protein, I have to tell the other cores the new position of the atom (because I have to update the interactions).
So, in each iteration I have to communicate to all the computers where the new position for all the atoms is.

So, this does not work, it is not a good approach.

An alternative that is used is "**Spatial decomposition**".



Instead of separating the work on a fixed set of atoms for each of the computers, we will separate the space in a fixed set of regions.

Let's say that you are in a plane in BCN, so you are controlled by the BCN tower. If you go to the North, how long do you think BCN tower will take care of you? Until you reach the border and they will tell the other tower that you are entering their zone.

It happens the same in our case. Each core is in control of a region of the space.

MPI → Takes control of X planes
Spatial decomposition → Takes control of a region (all regions have the same area, the image is misleading).

When an atom is going to cross the border, the core is going to communicate to the next one and it is going to transfer its "responsibility". So we only need to communicate the position of the atoms to the neighbors.
So, we are limiting the communication to the neighbors!

This is probably the major improvement. But there are also other improvements → GPUs and ASICs

HW optimization tricks: GPU, ASICs

A CPU is responsible of the main algorithm and works sequentially.

GPUs have become popular in simulation field.

- SIMD (Single instruction, multiple data) approach.
- One-two orders of magnitude faster that comparable CPUs
- Requires reformulate algorithms (you have to do a new code)
- On top of this, you can parallelize.

CPU is responsible of the main algorithm, GPU performs selected parts

- However, Amber is moving everything to the GPU!

ASIC: They are processors that are made on purpose to do MD (the best)

- The fastest ever MD computer (3 orders of magnitude above others)
- Guy that became rich and created a company with people that are really good in MD and in the creation of processors.

Other MD's

Coarse-grained

Aims to simplify either the representation of the molecules or the representation of the potentials. It reduces degrees of freedom to allow longer time steps.

Simplified representations: Instead of using atoms, we use particles (beads) that represent a number of atoms.

- MARTINI: Rule of 4. A bead corresponds to 4-atoms
  So we have 4 times less atoms, meaning that we wont need that much time.

This could be done at a larger scale, using beads of 10.000bp or even whole nucleosomes. This is done in big chromatin simulations.

Simplified potentials:

- Go-potentials: The interaction is 0 or 1.
  Assumes that native structure is a minimum. Interaction potential is 0 for atoms not in contact and negative for atoms in contact.
  - Used for folding or big conformational changes (big processes in which we are not concerned about the details)

At the end, we are reducing the number of degrees of freedom:

- Less atoms because you want to calculate things faster
- Less movements because you are not interested in them.

Analysis of simulations (of an experiment)

The value of a simulation comes from the analysis phase.

BIG PROBLEM, the simulation has evolved and improved (parallelizing, GPU…), but the analysis hasn't. We are using the same software as 10 years ago and our simulations are 10 times larger.

- Analysis should address anything that we measure in the trajectories/results (as if simulation was a real experiment) to solve the biological question.

At the end of the simulation we have a trajectory, a collection of conformations of the system over time (proteins, waters…), it's like a movie. Now we need to extract the information from it.

- Trajectories are "stochastic", which means that if we repeat a simulation, we will get different results, similar but not the same. We will not repeat the same simulation, and we will use an ensemble of averages of simulations for the different conformations
    - Exact position values change from one simulation to another
    - Analysis results come from "averages", "ensembles"

The reason why all trajectories are different relies in the fact that, the first thing we do is to apply random velocities to the atoms. since computers do not know how to do random numbers, if I use the same seed I will get the same random number.

To give atoms different velocities for the simulation we must set different seeds for a random velocity generation in the computer.
So having good representations of the same protein in different conformations will provide a good distribution and a good average result

Trajectory manipulation

Removal of PBC ("imaging")
First mandatory operation: go back, get rid of the waters and get the protein conformations to understand all the positions and coordinates we got in the simulation.

- PBC is removed by placing all system components in the "base" cell

Another thing we need to do is the **removal of the center of mass movement** (put the protein at the same place after each iteration).
Even if we try to keep the proteins in the same place, the sum of the velocities will give 0, which gives noise. Two ways:

- Can be done at simulation time, we can tell the simulation to remove it while it's calculating; but usually repeated before the analysis. As we don't trust the software much, we do it at the beginning of the analysis . Then we superimpose all the copies of the protein to the first one. But first, we need to remove the waters or it will be a mess

- Sometimes you need to analyze this movement. This is one of the trickiest things you can do in MD because MD models were made for static structures.
  If your biological problem implies diffusion, the protein should move.

- As mentioned, the removal of the center of mass is done by RMS, like in superimpositions with the first or average structure measurements. This can be done in 2 ways:
  - When you do a normal superimposition, you use **standard RMSd**: All atoms have the same weight. If the protein is a nice ball, nothing happens, but if the protein has a region that is really mobile, the superimposition is going to be very bad because in each simulation we will have a different conformation.
  - **Gaussing RMSd:** Non-moving atoms have a larger weight.

Finally, we do the removal of solvent/ions to reduce computation time:
- Unless any kind of solvent analysis is to be performed
- Requires also to generate a "dry" topology

Geometric analysis to perform quality control

Then we do a QC and this comes from geometric analysis. The idea is to make sure that the trajectory makes sense. You can do this by comparing it with experimental structures.
- We need to be able to reproduce the structure of the PDB. Obviously, one is a crystal and the other one is in solution.
  So, it is a good idea to do another simulation with the exact conditions of the PDB.

The things that we measure are:
- RMSd, Radius of Gyration, Native contacts (what is in contact in the PDB is still in contact?), secondary structure, changes in the surface.
  We will not get the same as the PDB, but the average should be close enough.

What we consider in geometry analysis
- Distances, angles, dihedrals, Hbonds
- Domain movements

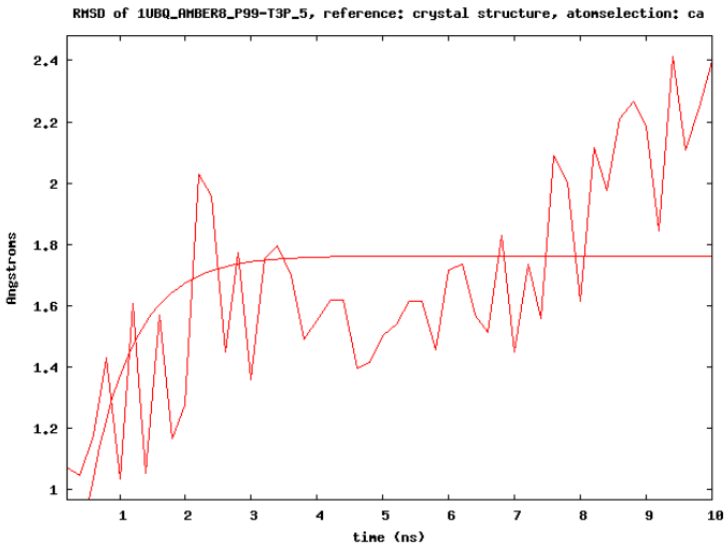
Trajectory stability

We must check the stability of the trajectory.
Ideally, any of the measures we were using in QC are good for this. The most common one is RMSd.
Equilibrium trajectories should keep geometric values stable
Given the fact that the distance between the experiment and the average is less than 2Å, the calculation of the RMSd must be between 2-3Å that distance.

Non stable trajectories may reveal
- Incomplete equilibration and therefore you need to go back to the set up and do a better equilibration step.
- Structure is evolving to new states (folding, unfolding, fit), so maybe the structure is correct.
- Fluctuation among states

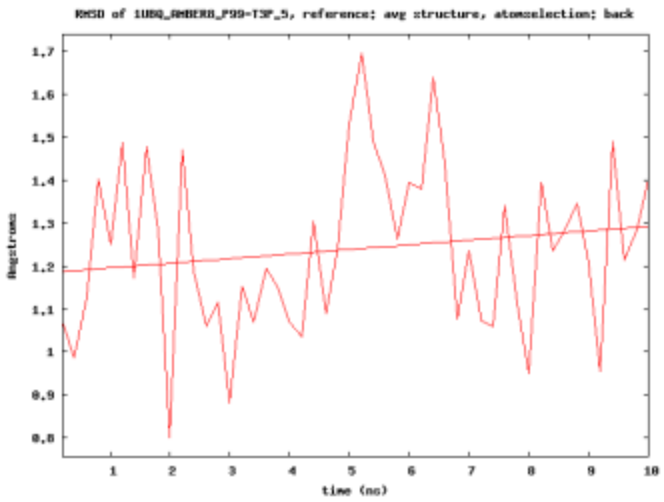
Use the experimental structure to plot all trajectories and find the most stable conformation. As we can see there are some fluctuations and the line they form isn't flat; so the plot doesn't represent a stable enough conformation.

We are also moving away from the original structure.



1ubq 20ns (from exp)

Taking the average as the reference should be fine in theory; because it's the average, but as we can see the plot isn't flat, so the average doesn't represent all the fluctuations. Even though it's not perfect, it's the most used option.



1ubq 20ns (from avg)

Having fluctuation among the states is normal, the simulation isn't wrong; but having this amount of fluctuation requires the process of clusterization…

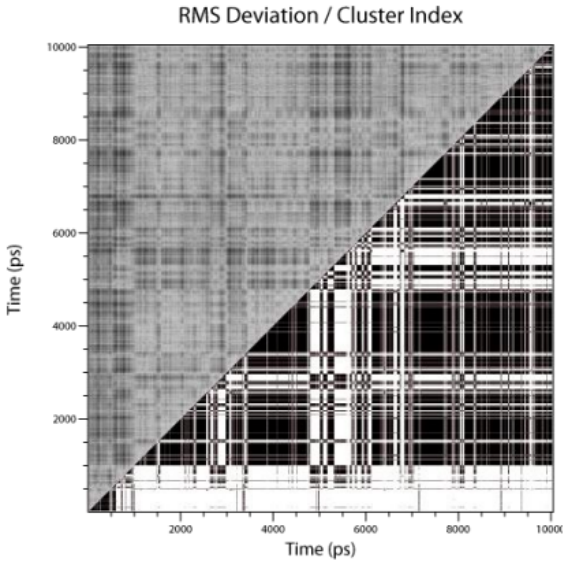
Clusterization

The principle is the same as in the plots; but this is more specific to find particular states. The interpretation of the cluster is the relevant part. Trajectory may reveal the existence of several **states;** each cluster corresponds to a different state or conformation, and their transitions in between.

If the clusterization works; we should find many transitions and some individual states. Ideally, the trajectory should show reversible transitions.

• Clusterization algorithms seek to identify individual states, by comparing distances between snapshots. Typical "distances" (mathematical distance in the clustering world):

- Our meaning of this distances refers to RMSds or an average of the internal coordinates. (anything able to separate one state from another)
    - RMSd (on coordinates or intramolecular distances)
    - Geometric features



• Algorithms

- Single linkage (distance to cluster members less than cutoff)
- Maximum number of neighbors (close structures)
- "Gromos" Iterative max neighbors
- K-means, …

Clusters are represented by its "centroid" structure or by averages

The usage of a cluster could reveal that in a collection of conformations of a protein we could find a series of transitions and just 3 states to study, so 3 pictures or representations of the protein. Then making a movie with the states we can see the changes in the conformation and highlight the differences between the states.

To find the likelihood for each confirmation we use Markov State Models. Takes into account the transition states as the transition probabilities and the actual states to calculate how likely the conformation will form.

We have a number of states in our system and each state is more or less populated, so it is a measure of likelihood (Boltzmann). So, we can just check which states appear more often.

We also have transitions between states. The number of transitions between 2 states is going to give us the probability of transition.
This creates a network of transitions connecting states that is studied by MSM.

Markov State Models (MSM)

It is made of a combination of results of several simulations (done in different computers → parallelizing) that are joined together.

• How?

- 1 st define states (usually from cluster analysis)
- 2 nd: Once we have the states, we go to the trajectory and measure the probability of transition between states (Transition matrix)

As the transition matrix is much more stable than the populations, we do not need a long simulation to get all this. With a series of short simulations, you can get a good transition matrix (it converts much faster).

So, using this approach, we can actually predict the probability of evolving in some specific trajectory.
We get a good picture about what is going to happen in the next millisecond of simulation.

Flexibility analysis 1D

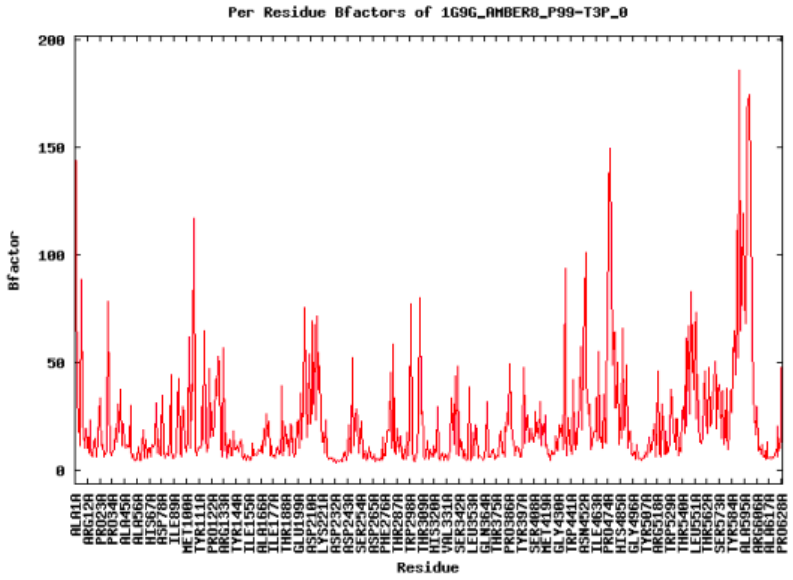
MOST IMPORTANT THING OF THE ANALYSIS → Getting the dynamic information, how does the protein move.

The simplest version is 1D (one direction) calculating the RMSf.

- RMSd is the deviation from one state to another
- RMSf is the deviation from the average (Atomic fluctuations). It's the fluctuation around the central value

$$RMSf = \sqrt{\langle (r - \langle r \rangle)^2 \rangle} = \sqrt{\frac{\sum_{Snap}(x_i - x_i^{avg})^2 + (y_i - y_i^{avg})^2 + (z_i - z_i^{avg})^2}{N}}$$

If we compute this along all the trajectories, we get a plot where we have the residue vs fluctuation:



This is the typical pattern.
If you think of a normal protein, the center is going to be stable and the terminals are going to be less stable (they have loops).
The plot represents which are the parts of the protein that are more mobile.

RMSf translated into B - Factors
That column in the pdb that was understood as fluctuation; because the B factor is actually a measure of the fluctuations.

Flexibility analysis (2D) Covariance matrix

I make a covariance matrix that contains all 3N x 3N covariance values (all coordinates against all other coordinates). N = number of atoms

- Main diagonal values of the matrix give atomic RMSf's or variance of each atom
- Non diagonal values are correlations.
- The inverse of the covariance is the correlation!

The information that I get is which parts of the protein are moving together. I may have a protein with a big domain that is moving all together and therefore I expect a big correlation between the atoms in the same domain and a low correlation with the rest of the protein

So, I can get collective movements → Groups of aa that are moving together. This helps me to understand conformational changes, allosteric proteins…

Principal component analysis (Essential dynamics)

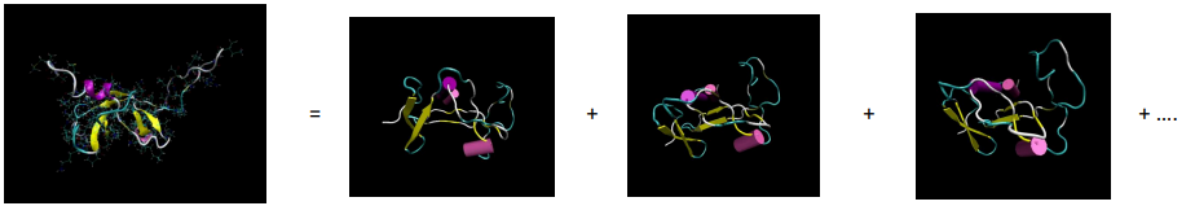
• Allows split system movement (decomposing the movement) in its major components and perform separate analyses with eigenvectors and eigenvalues to use statistics and study the trajectories.

So, we can understand something that otherwise is very complex.

The point is that we obtain the natural movements → We obtain the major part of the variance(principal component) of the movement of the protein. This represents the principle conformational changes that our protein will make.

So, it's a wonderful way to extrapolate the fluctuation and therefore generate new conformations.

• 70-75% of the variance is usually achieved with less than 10 components.



Since we sort them according to their importance, the first ones are the ones that cover the largest variance. At the end we obtain the weak movements that are not relevant for the conformation.

Entropy calculations

Entropy values can be deduced from PCA analysis using pseudo harmonic approaches

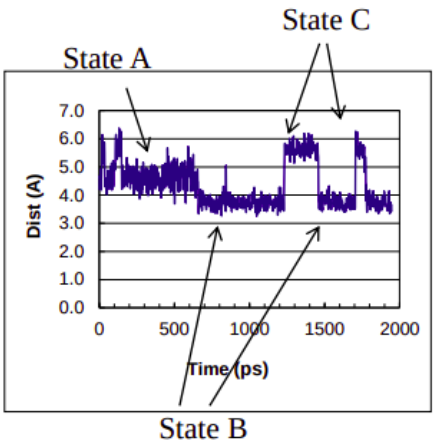
Energy analysis

If I have 3 states and I count how many times I have each state, I can then compute the probability of transition between each state.

It's a way of putting numbers to plots, referring to populations for each state

Energy plots along simulation allow to check trajectory stability, energy conservation

- The thermodynamic values refer to how many snapshots or counts per state
- The arrows are the kinetics, the transitions i have between states



• Binding/interaction analysis

We know how to calculate energy between 2 molecules in a complex; if instead of having 9 conformations we have a lot more, we use an average of binding energies with the trajectory vectors (to include entropic fluctuation):

- Interaction energy is averaged through the trajectory (weighted by population giving free energies).
- ELECTROSTATICS: we can calculate the interaction with water because it is in the simulation, so we also use an average of the interactions between the molecule and the waters
- MM-PBSA, MM-GBSA methods average interaction energies in gas-phase (from MD) which the program of the simulation already gives; electrostatic solvation (PB/GB, 2 methods), and hydrophobic solvation (SA, surface analysis)

This is 1 of the methods to get the binding energies along simulations. The point here is to use the average.

ML & AI

• Initial experiences in applying artificial intelligence to the MD analysis
  - Mostly used: dimensionality reduction (Flexibility analysis, PCA-like)
  - Force field parameterization
  - Dynamics reaction coordinates, knowing in which direction the protein is going to move
  - Synthetic trajectories, like the MSM do
  - …

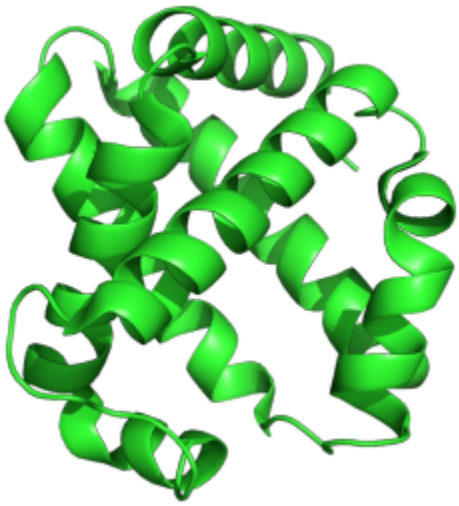
Practical 8. Molecular dynamics analysis

Step 1: Quality control

To assess the quality of a molecular dynamics simulation we check that the simulated protein structure remains stable during the whole simulation. To check that we have two measurements: the RMSD and the radius of gyration.

We saw the concept of RMSD in practical 3 when we were talking about superimpositions. It is a measure of how different are the locations of two sets of atoms in space. If we apply this concept to proteins, it tells us how different is the location of atoms in space between two proteins. When we apply the RMSD in a molecular dynamics, we use it to compare our structure during the simulation with the structure at the beginning of the simulation. The structure at the beginning of the simulation can be the original structure from the PDB, the structure after energy minimization or the structure after equilibration.
The concept of radius of gyration is a measurement of mass density in the protein that we are studying. It is defined as the root mean square distance of the object's parts from its center of mass. To better understand this concept, check the following examples:

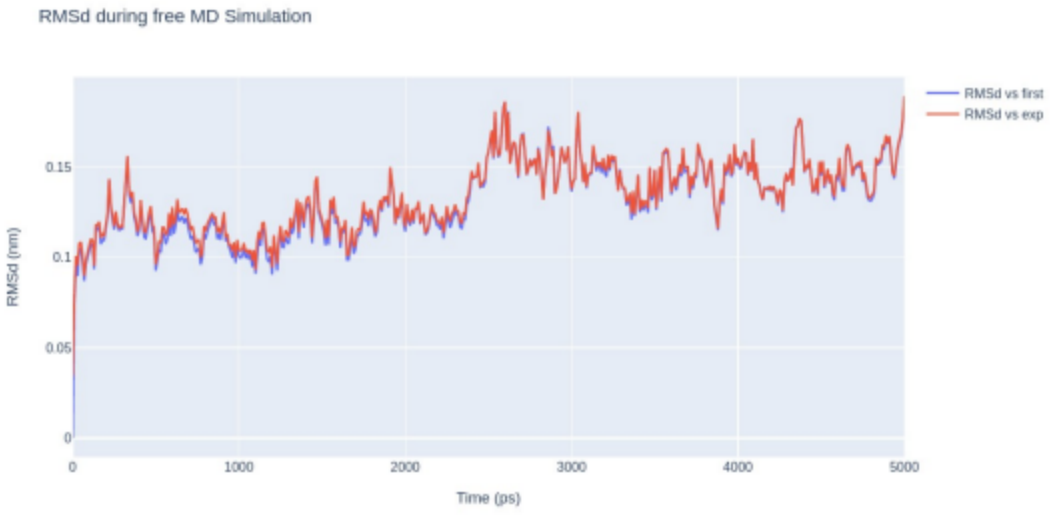
- If a protein is highly compacted its radius of gyration will be small, because there will be a short distance between its center of mass and its peripherical regions. A globular protein such as hemoglobin are one example of compacted proteins.



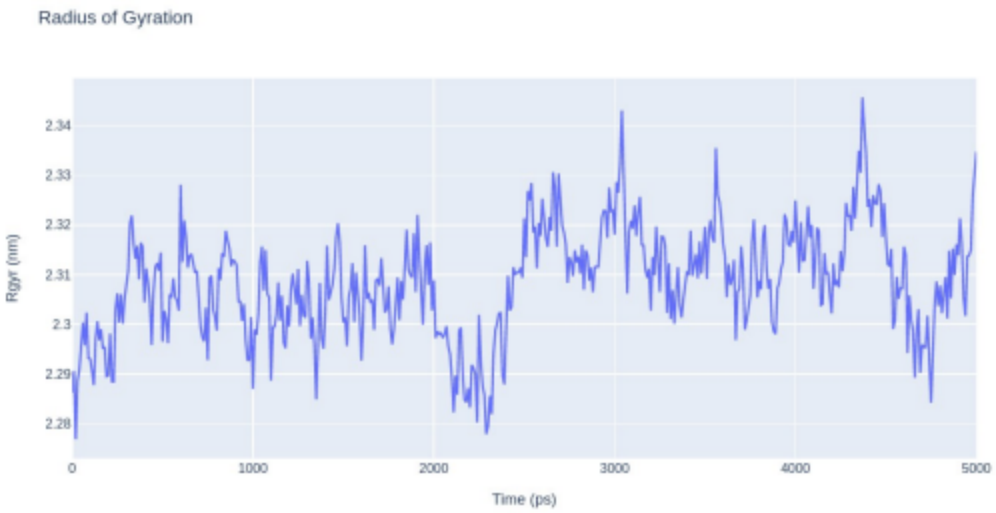
- If a protein is very dispersed its radius of gyration will be big, because there will be a long distance between its center of mass and its peripherical regions. Unfolded proteins are one example of very dispersed proteins.

For the RMSD and the radius of gyration we evaluate stability over time. Usually when the dynamic starts these values change from the initial state (which is just after equilibration). However, after this first sudden change we expect them to remain stable. Sudden or continuous changes in these variables may indicate that the dynamic made the protein to adopt a conformation that is far away from its native state. When this happens is mostly because some of the parameters of the simulation were not adequate, rather that indicating that the protein can adopt this new conformation spontaneously.

Next you can see the plot for the RMSD comparison between the different frames of the dynamic and the initial structure. See that we are having two lines, one considering the initial structure the result of energy minimization (blue) and another considering as the initial structure the one of PDB (red). We see that the RMSD is slightly lower when we use the energy minimization structure as a reference. This makes sense because energy minimization helps the system to find low energy conformations. Molecular dynamics usually also bring the protein to low energy conformations. Therefore, the structure resulting from energy minimization is more similar than the PDB structure to the conformations that appear along the simulation.



Next you can see the radius of gyration. In this plot we see that there is some variability, but check that the axis of the plot involves a really small range of values. Therefore, this plot also shows that the protein remains stable during the whole simulation:



Step 2: Visualize your trajectory using pymol

Another way to check if your simulation went alright is to visualize it and see what conformations your protein adopts. Some errors in the simulation, such as protein unfolding, can be seen easily just by watching the simulation.

To load the dynamic we have to follow the next steps:

1. Open pymol.
2. Load the gromacs file (.gro) generated by your free molecular simulation.
3. Load the trajectory file (.trr) generated by your free molecular simulation. Select the default parameters that you are asked when you load the dynamic.

Now you will see in your screen your protein, the water molecules, the ions and lines describing the movement of water molecules. To simplify the image you are seeing type in the command line:

hide lines

Now you will see only the protein and the ions. If you go to the menu you have in the lower right corner of the screen and you click on the start button (triangle pointing to the right) you will visualize your molecular dynamics trajectory. To stop the trajectory click on the stop button (square).

When I visualize my trajectory with pymol I can see 501 frames. If this trajectory corresponds with 5 ns, how much time corresponds with each frame? Apply this questions to your trajectories in case that you have different trajectory times or different amount of frames.
If you remember, we imposed that the simulation should have 2500000 steps. If pymol shows 501 frames (also called states by pymol) it is clear that each frame does not correspond with each step. Instead of that, each frame corresponds with 5000 steps, and each step corresponds with a time interval of 2 fs. Also, keep in mind that the first step

corresponds with the structure before the molecular dynamic starts. So in reality, our dynamic has 500 frames, plus the first one which is the result of equilibration.
If the trajectory moves too fast, you can slow down the pace of the trajectory by changing the frame rate. To do this go to movie, then frame rate, then select a lower frame rate.

movie > frame rate

You can save your pymol visualization of the trajectory as a movie, and you can include this movie in your presentation if you want. To do this go to file and to export movie as.

file > export movie as

See that if you edit the visualization of some parts of the protein, maybe by showing sticks or by changing colors, this changes remain along the simulation. The changes in visualization will also be included in the movie that you generate at the end.

Step 3. Measuring distances and monitoring contracts along simulations

Measuring distances between specific atoms is one of the types of information we can retrieve from molecular dynamics. They are important because they allow us to monitor important interactions in the protein and how this interactions evolve during time. Also, by checking for how long an interaction takes place during the trajectory, we can infer the relevance of that interaction for the stability of the protein.

Take as an example the case of one hydrogen bond. Hydrogen bonds are established between oxygen or nitrogen atoms that are at a maximum distance of 3.5 angstroms. Also, one of the two atoms must be covalently bonded to a hydrogen atom. It could be the case that you find a hydrogen bond in the static structure from the PDB, and you may think that this hydrogen bond contributes to the overall folding and function of the protein. But it could be the case that this hydrogen bond is an artifact from creating the PDB structure, or that in some conformations of the protein it doesn't exist.

To further prove the role of a hydrogen bond, you can monitor the distances of this bond along the trajectory. If the distance between the implied atoms goes beyond 3.5 angstroms you know that the hydrogen bond is no longer happening. Here you could find 3 different scenarios:

- The hydrogen bonds stays under 3.5 angstroms of distance for the whole simulation. In this case the hydrogen bond is always there and it clearly proves to be involved in in maintaining some fundamental regions of the fold of the protein.
- The hydrogen bonds stays under 3.5 angstroms of distance for some parts of the simulation, while for others it takes longer distances. This means that, if the simulation works correctly, this hydrogen bond is involved in establishing some of the conformations that this protein can adopt. So, depending on what conformation the protein is the hydrogen bond can be there, or not.
- The hydrogen bond only stays under 3.5 angstroms of distance for the first frames of the simulation. This means that the hydrogen bond was an artifact from the crystallographic structure. When proteins are crystallized sometimes they adopt conformations that are not exactly the native conformation. Molecular dynamics puts this protein in a physiological environment, where the protein can easily recover its native conformation. Then, if the hydrogen bond is lost at the beginning of the

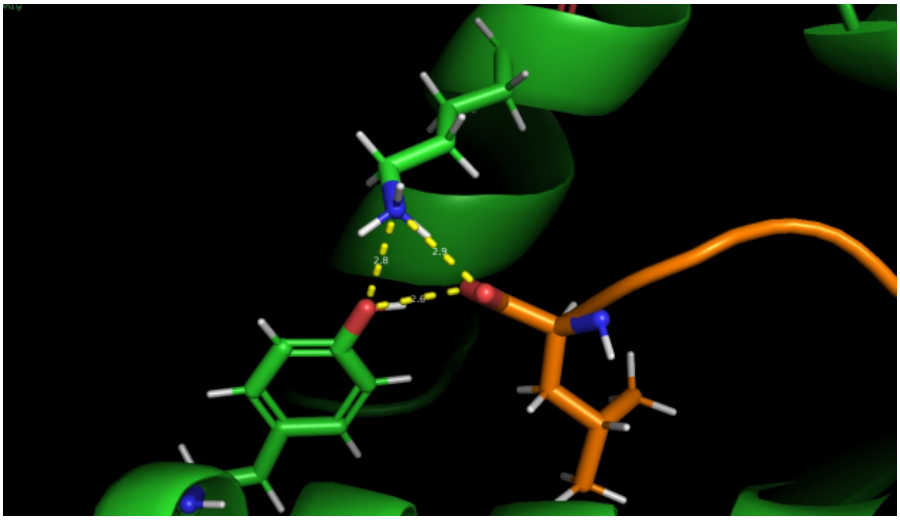
simulation and never appears again, it is quite likely that it was an artifact of the crystallization.

Knowing this, now you have to explore your proteins. Find some contact involving amino acids that it is important for your protein's function (the contact can be between amino acids or between amino acids and other molecules such as nucleotides or drugs).
Once you have decided what contact you will work with, you can measure the distance in this contact with pymol. You have to click on wizard and use the measurement tool. The measurement tool will open a new window in the right margin of the screen, this is how it looks like:

Then you have to click on distances and finally click on the atoms whose distances you want to assess. You can choose as many pairs of atoms as you want.

wizard > measurement > distances > click on atoms

In the simulation of the MHC-I (the one that I am providing to you) I have chosen the contacts between 3 amino acids in order to calculate distances. These amino acids are TYR 84 and LYS 146 from the heavy chain (green) and LEU 9 from the viral protein (orange). Here you can see the distances between these amino acids at the first frame of the MHC-I simulation:



Now you have labeled the distances for one or more contacts. If you start visualizing the different states of you simulation you will see how this distance changes from frame to frame. However, checking the distance frame by frame is tremendously inefficient and tedious. That is why we will use a pymol script to record these distances at the different frames of this simulation.

One of the great advantages of pymol is that you can execute python scripts within itself. These scripts include many functions that are exclusive for pymol such as loading structures, loading trajectories or measuring distances between atoms. In the moodle you have an example for a simple pymol script that you can use to measure distances between some atoms for all the frames in your simulation. Then, these distances are written in an output file that you can easily parse to create plots.

Use this script to measure the distances between the atoms that you are interested. You will see that the paths and the names of the atoms in this script are made to match the molecular dynamics trajectory I created in my own computer. You have to change the path of the files and also change the atoms whose distance you want to measure. Also, in this script I am computing 3 distances. Feel free to change that and compute as many distances as you like, but at least monitor one distance. To identify the atoms, if you click in an atom in pymol you will obtain an identifier for that atom in the pymol console, it is highlighted in red in the following image:

Modify the script and put the paths to files and atoms that you want to use in your analysis. Then, execute the script by running in a terminal:

pymol write_distance.py

If this command doesn't work and you have pymol located in one specific directory of your computer, go to that directory and execute:

./pymol "path to the write_distance.py script"

If everything goes fine, a new file should appear in your current directory and it will include distances for all the molecular dynamics frames that you can see with pymol. Also, since you know the amount of time that corresponds with each frame, you can know at what time of simulation each of the distances are been measured.

Frame	Time(ps)	Distance_1	Distance_2	Distance_3
1	0.0	3.4552156925201416	2.834542989730835	3.347724676132202
2	10.0	5.227877616882324	3.1723783016204834	3.6019208431243896
3	20.0	3.528360605239868	2.807157039642334	3.3302125930786133
4	30.0	3.598281145095825	3.2178215980529785	4.280811309814453
5	40.0	3.2978575229644775	3.3889284133911133	3.9164106845855713
6	50.0	3.3590707778930664	2.9832990169525146	3.3365743160247803
7	60.0	3.393904447555542	2.976013660430908	3.515756607055664
8	70.0	3.3602101802825928	2.9066059589385986	3.463763475418091
9	80.0	3.3225433826446533	3.0552687644958496	3.417757511138916
10	90.0	3.462924003601074	2.8074405193328857	3.286269426345825

Finally, use python or any other program you want to make plots of how these distances change across the dynamic and hypothesize what is the role of the interactions you are studying in this exercise.

Step 4. Energy analysis

So far we have focused on distances, atoms and conformations, but we are missing a key point of the molecular dynamic: energy. Remember that on each step of the simulation, the force field is measuring the entire energetic landscape of our chemical system. We can get a glance of the changes in energy across the simulation by checking the log file. The log file contains energy values for the whole simulation every 5000 steps. These packs of 5000 steps also correspond with the frames of the simulation that we see when we load the trajectory in pymol. The log file also provides values about the average energy values of the simulation and the computation resources involved on it. The problem with the .log file is that it is hard to parse because it contains the information in a very unorganized way. That is why we have the .edr file, which contains all the information regarding energy, temperature, pressure and other physicochemical properties of the simulated system. The .edr file is a binary file, this means that we cannot see the information that it contains because is encoded. However, we can use the biobb package to extract the contents of this file and include them into a new file which will be very easy to parse. To do this follow the next workflow: Start by loading the conda environment corresponding to the tutorial we started last week:

conda activate biobb_MDsetup_tutorial

Now execute python and from the python console you can extract the information inside the .edr file with a couple of commands:

python

Load the package , which is used to extract information from .edr files:

>>>from biobb_analysis.gromacs.gmx_energy import gmx_energy

Define a dictionary with the variables from which you want to get information. In this example we are extracting information from the potential energy of the system, the temperature of the system and the pressure of the system.

>>>prop = {'xvg': 'xmgr', 'terms': ['Potential', ‘Temperature’, 'Pressure']}

Finally, execute the command to extract this information and store it into a new file:

>>>gmx_energy(input_energy_path=”path to .edr file”, output_xvg_path = “path to an output.xvg file’, properties=prop)

The .xvg file will contain in a simple format the requested variables for each frame of the simulation. Remember that each frame corresponds with 10 ps.
This is how the output .xvg file should look like:

```
# This file was created Thu Mar  3 17:34:38 2022
# Created by:
#                     :-) GROMACS - gmx energy, 2019.1 (-:
#
# Executable:   /home/albertosaurio/anaconda3/envs/biobb_MDsetup_tutorial/bin/gmx
# Data prefix:  /home/albertosaurio/anaconda3/envs/biobb_MDsetup_tutorial
# Working dir:  /home/albertosaurio/pymol
# Command line:
#   gmx energy -f /home/albertosaurio/molecular_dynamics/MHCI_md.edr -o /home/albertosaurio/molecular_dynamics/MHCI_md.xvg -xvg xmgr
# gmx energy is part of G R O M A C S:
#
# Gnomes, ROck Monsters And Chili Sauce
#
@    title "GROMACS Energies"
@    xaxis  label "Time (ps)"
@    yaxis  label "(kJ/mol), (K), (bar)"
@TYPE nxy
@ view 0.15, 0.15, 0.75, 0.85
@ legend on
@ legend box on
@ legend loctype view
@ legend 0.78, 0.8
@ legend length 2
@ legend string 0 "Potential"
@ legend string 1 "Temperature"
@ legend string 2 "Pressure"
    0.000000  -1248924.625000  300.141846  -44.758331
   10.000000  -1248278.750000  299.768890   98.174606
   20.000000  -1248357.000000  300.726013   38.976257
   30.000000  -1248233.625000  299.345154  -21.266464
   40.000000  -1247402.375000  300.384613   39.322968
   50.000000  -1248242.500000  299.854614  -296.071533
   60.000000  -1248758.125000  301.414337  209.375000
   70.000000  -1249095.125000  299.513855  -102.707840
   80.000000  -1251865.375000  300.696777   95.248993
   90.000000  -1250915.375000  299.329895  -79.612289
  100.000000  -1250344.500000  300.213318  134.949768
```