# Exercise 4: Modeling protein structures through comparative modeling and threading.

We propose the following problem: we want to study a protein for which we know the sequence but there is no available structure. To perform a complete study of the protein we will obtain models of the protein structure. This situation is very common because high-throughput technologies have produced a vast amount of protein sequences, while the number of high-resolution structures has seen a limited increase. In this tutorial we will learn how to obtain structural models from protein sequences using two different approaches: comparative modeling and threading.

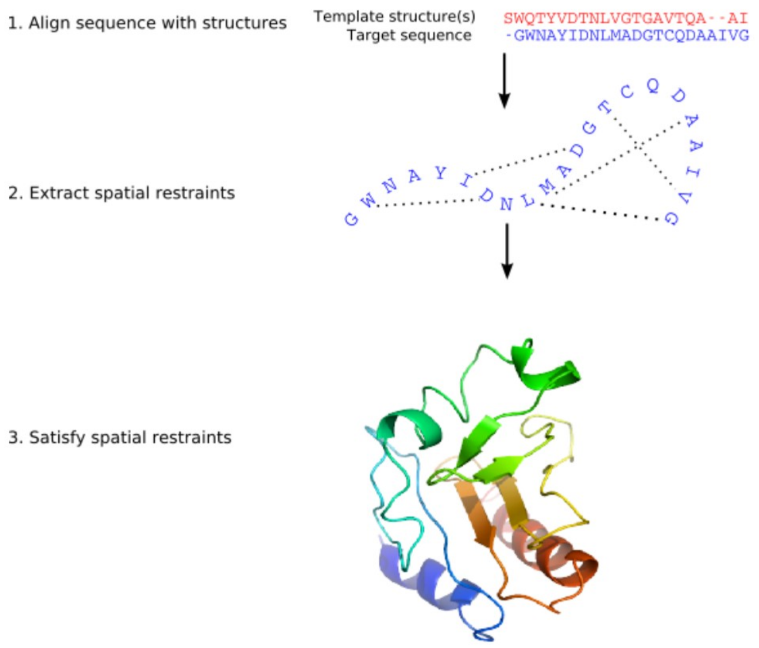
## Theoretical concepts:

**Comparative modeling (or homology modeling):** it is the process of modeling the structure of one protein using as template/s the structure/s of homologous proteins. This process is based on the next facts:

- The number of protein folds in nature is limited. Many protein structures are obtained every year, but rarely these structures have an unknown fold.
- Proteins with similar sequences have an almost identical structure.
- Homologous proteins tend to share the same protein fold.

There are several programs to perform comparative modeling, in this tutorial we will learn how to use MODELLER. MODELLER is a program that models protein structures based on the satisfaction of spatial restrains. It works following the next steps:

1. The sequence of the target and the sequence of the template are aligned. This means that each residue of the target is assigned to one residue in the structure of the template.
2. Spatial restrains, such as Ca-Ca distances, are transferred from the templates to the target. Besides, probability density functions are obtained from the structural features of the templates.
3. A 3D model is obtained by satisfying all the spatial restrains as well as possible and optimizing probability density functions and energetic terms.



Using comparative modeling we can obtain many different models for the same target. Some of these models will be more accurate than others. These are some variables that can influence in model accuracy:

- The homology degree between the target and the template. The highest the homology, the highest the accuracy of the model.
- The quality of the alignment between the target and the template. If the residues of the two proteins are not correctly aligned the model is prone to be inaccurate.
- The optimization of the model. After applying the spatial restrains, MODELLER performs a fast simulation in order to optimize the location of all the atoms in the model. This process can produce different outputs. That is why MODELLER can produce one or more different models from the same template.

This rises the fact that we need some mechanisms to assess the quality of the generated models. During this lesson and in the following one we will explore two ways of evaluating the quality of models: the use of statistical potentials and the prediction of secondary structures.

**Threading:** is the process of modeling the structure of one protein by assigning to that protein an already known fold. It works by fitting a protein sequence into an already known structure. This strategy has not as good results as comparative modeling, but it can be a useful resource in case you want to model one protein for which you find no templates.

Threading programs typically have a database of protein structures. So, in stead of making one model they make a lot, using their different available structures. Thus, here we have the same problem that we found in comparative modeling: we have a lot of models and we should differentiate the accurate ones from the rest. One way to do this is to use statistical potentials.

**Statistical potentials:** statistical potentials are probability functions derived from the analysis of a reference set of known protein structures. This function will provide good scores for the protein models that have similar structural properties to the ones in the reference set. Then, we make that the structures in the reference set are in a native state. By doing so, good scores will tell us that a protein model has similar structural features than our set of native state proteins. These good scores are telling us that our model is accurate and reliable. Keep in mind that low scores are good scores, and high scores are bad scores. So, the lowest the best.

This function is obtained by calculating the inverse of the Boltzmann law using the probabilities found in the reference set. This allows the representation of probabilities as energy values. Therefore, statistical potentials scores can also be called energies. Low scores will involve low energies and high similarity of the assessed model with the native structures in the reference set.

$$P=(1/z)(e^{(-E/kT)})$$

$$E= -KTlnP + KTlnZ$$

The structural properties that are taken into account by the statistical potentials that we will use today are:

- Distances between amino acids. This distance can be computed between the two closest atoms of each residue pair or between specific atoms of the amino acids. In the second case, it is common to compute the distances between the beta carbons, because by doing so we take into account the orientation of the side chains. In most programs this feature is used to compute the “PAIR” energy.

- Degree of exposure (or solvation) of amino acids: amino acids can be placed in the core or in the surface of the protein. We expect to find hydrophobic amino acids in the core and polar amino acids in the surface. In most programs this feature is used to compute the “SURFACE” energy. Solvation stands for the degree of interaction of amino acids with water molecules. The most exposed an amino acid is, the more water molecules it will interact with.

It is important to keep in mind that statistical potentials are relative measurements. This means that we cannot apply a score threshold to discriminate good from bad models. They have been intended to compare models between them or with experimentally determined structures.

A good strategy to assess whether a protein has been correctly modeled is to compare the statistical potentials energies between the model and its template. We expect the template to have a good energetic profile, since it is an experimentally determined structure. Therefore, we can use the template as a reference. If the model has higher energies than the template, it is very likely that it has been wrongly modeled. This same strategy can be used to identify wrongly modeled regions inside a model, instead of evaluating the whole model at once.

## Tutorial:

### Step 1: Find templates

Move to the “MODELLER” folder inside the “exercise_4” directory. We will obtain models for the sequence contained in the file “P11018.fa”.The first step in homology modeling is always to find good templates for our target protein. During exercises 2.1 and 2.2 we learnt how to use BLAST and Hidden Markov Models to do this, and we explored different strategies to find reliable templates. Today we are going to search for templates in a very simple way for the sake of time. We are going to run BLAST into the PDB database with our target sequence using the following command:

```
blastp –query P11018.fa -db ~/Documents/databases/pdb_seq
-out P11018.out
```

From the BLAST output we can retrieve the PDB IDs of the proteins that we are interested to use as templates. We can use one or more templates to model our target protein. With the PDB IDs we can get the structures of the templates from the PDB database.

1. Go to the PDB website in: https://www.rcsb.org/

2. Copy the PDB ID of the template in the search window and click enter.

3. Go to the PDB entry that has the template ID. Once in there, click on the download files button. Choose to download a file in PDB format.

By default, by doing this the PDB file of the template will be stored in your downloads directory. Copy the downloaded PDB file/s into the MODELLER directory.

It may be the case that from the whole PDB file we are only interested in one of the chains. Besides, PDB files can have parts that are not well understood by other programs. To solve these two issues we use PDBtoSplitChain. This program splits one PDB file in different chains and corrects the PDB format. It also provides us the fasta sequence of each one of the chains. Run the following command for each one of the templates you want to use:

```
perl ~/Documents/perl_scripts/PDBtoSplitChain.pl –i template.pdb –o prefix
```

**Step 2: Build a protein model**

We are going to obtain models for our target protein using a program called modeller. To execute MODELLER you need three files:

- Target file (target.fa): contains the target sequence.
- Alignment file: contains the alignment between the target and the template/s in PIR format.
- Script file (modeling.py): contains the executing commands for MODELLER.

We already have the target file and the script file in our working directory. So we will start by creating the alignment between the target and the template/s. In previous exercises we have seen many strategies to build alignments, in particular we learnt that alignments built using Hidden Markov Models (with hmmalign) have more quality that the ones produced by clustalw or other agglomerative alignment programs. For the sake of time we will build this alignment using clustalw. In the next practical lesson we will make this alignment using Hidden Markov Models and we will compare the results.

Now, we concatenate the sequences of our target and the template/s that we want to use by running the next commands:

```
cat P11018.fa > target_template.fa
```

```
cat template.fa >> target_template.fa
```

Now, we align these sequences using clustalw:

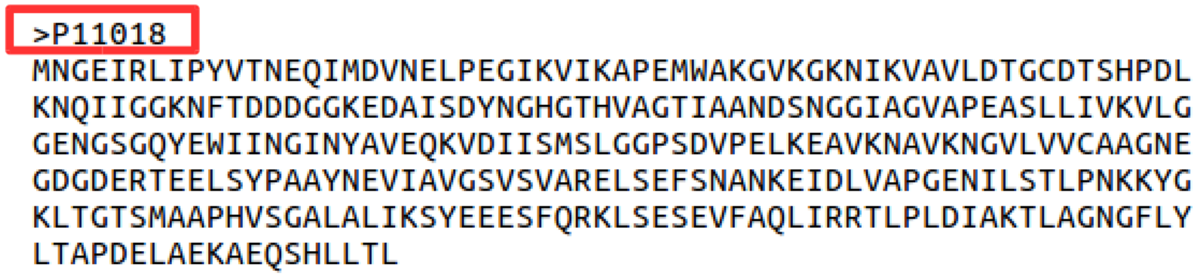
```
clustalw target_template.fa
```

We have obtained a MSA called “target_template.aln”. Now we need to modify the format of the alignment file, from clustal to pir format. We will do so by using aconvertMod2.pl:

```
perl ~/Documents/perl_scripts/aconvertMod2.pl –in c –out p
<target_template.aln>target_template.pir
```

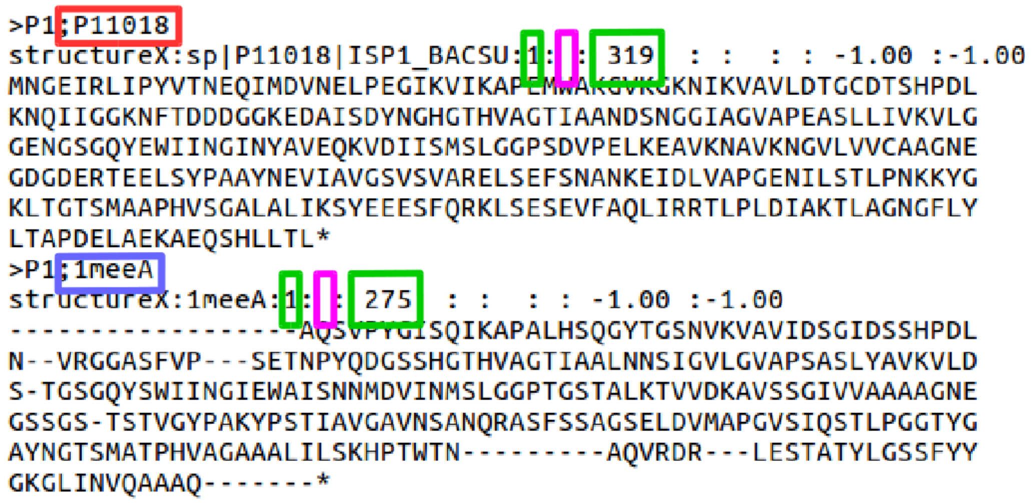
We must be very precise fulfilling all the requirements from MODELLER regarding the PIR alignment, otherwise the program will not work. MODELLER requires the target, the alignment and the script files to have the same names and codes regarding the files that will be handled. Here is an example of how these files should look like:

**P11018.fa**



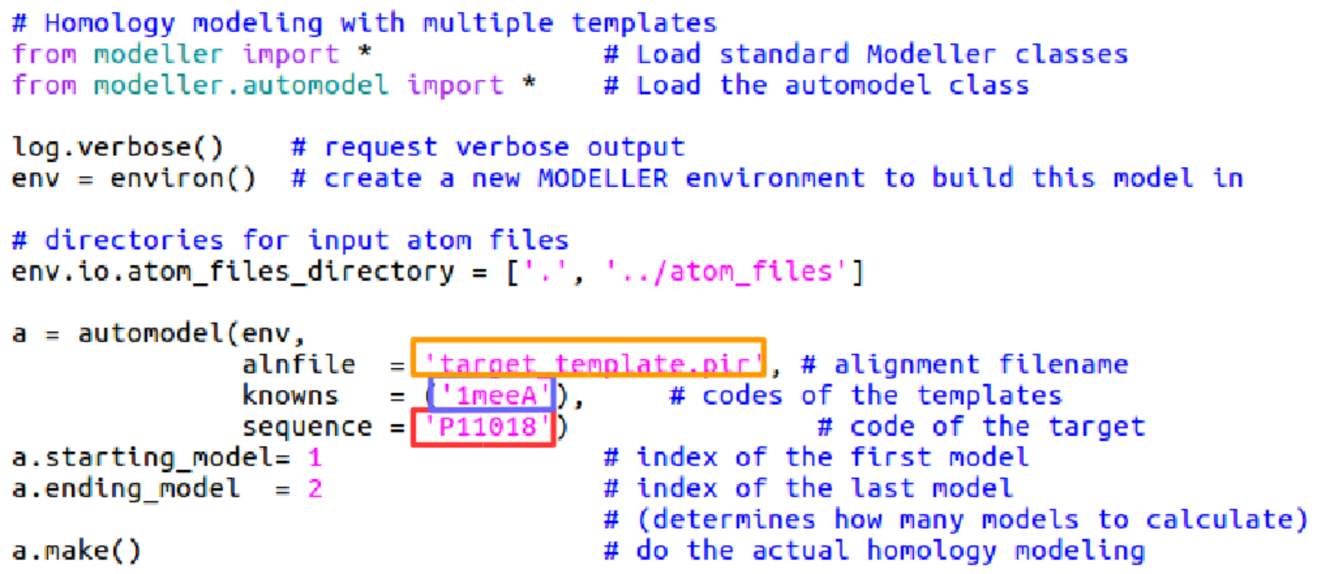
```
>P11018
MNGEIRLIPYVTNEQIMDVNELPEGIKVIKAPEMWAKGVKGKNIKVAVLDTGCDTSHPDL
KNQIIGGKNFTDDDGGKEDAISDYNGHGTHVAGTIAANDSNGGIAGVAPEASLLIVKVLG
GENGSGQYEWIINGINYAVEQKVDIISMSLGGPSDVPELKEAVKNAVKNGVLVVCAAGNE
GDGDERTEELSYPAAYNEVIAVGSVSVARELSEFSNANKEIDLVAPGENILSTLPNKKYG
KLTGTSMAAPHVSGALALIKSYEEESFQRKLSESEVFAQLIRRTLPLDIAKTLAGNGFLY
LTAPDELAEKAEQSHLLTL
```

**target_template.pir**



```
>P1;P11018
structureX:sp|P11018|ISP1_BACSU:1: : 319  : :  : : -1.00 :-1.00
MNGEIRLIPYVTNEQIMDVNELPEGIKVIKAPEMWAKGVKGKNIKVAVLDTGCDTSHPDL
KNQIIGGKNFTDDDGGKEDAISDYNGHGTHVAGTIAANDSNGGIAGVAPEASLLIVKVLG
GENGSGQYEWIINGINYAVEQKVDIISMSLGGPSDVPELKEAVKNAVKNGVLVVCAAGNE
GDGDERTEELSYPAAYNEVIAVGSVSVARELSEFSNANKEIDLVAPGENILSTLPNKKYG
KLTGTSMAAPHVSGALALIKSYEEESFQRKLSESEVFAQLIRRTLPLDIAKTLAGNGFLY
LTAPDELAEKAEQSHLLTL*
>P1;1meeA
structureX:1meeA:1: : 275  : :  : : -1.00 :-1.00
------------------AQSVPYGISQIKAPALHSQGYTGSNVKVAVIDSGIDSSHPDL
N--VRGGASFVP---SETNPYQDGSSHGTHVAGTIAALNNSIGVLGVAPSASLYAVKVLD
S-TGSGQYSWIINGIEWAISNNMDVINMSLGGPTGSTALKTVVDKAVSSGIVVAAAAGNE
GSSGS-TSTVGYPAKYPSTIAVGAVNSANQRASFSSAGSELDVMAPGVSIQSTLPGGTYG
AYNGTSMATPHVAGAAALILSKHPTWTN---------AQVRDR---LESTATYLGSSFYY
GKGLINVQAAAQ-------*
```

**modeling.py**



```
# Homology modeling with multiple templates
from modeller import *                    # Load standard Modeller classes
from modeller.automodel import *    # Load the automodel class

log.verbose()    # request verbose output
env = environ()  # create a new MODELLER environment to build this model in

# directories for input atom files
env.io.atom_files_directory = ['.', '../atom_files']

a = automodel(env,
              alnfile  = 'target_template.pir', # alignment filename
              knowns   = ('1meeA'),     # codes of the templates
              sequence = 'P11018')              # code of the target
a.starting_model= 1                 # index of the first model
a.ending_model  = 2                 # index of the last model
                                    # (determines how many models to calculate)
a.make()                            # do the actual homology modeling
```

The relevant elements of these files are highlighted in colors:

- Target name (Red): is the name of the target protein. It should be the same in the three files. It is written without the ".fa" extension.

- Template name (Blue): is the name of the template PDB. It is written without the ".pdb" extension on the .pir and modeling.py files. Besides, the corresponding file should be in the working directory.

- Start and end amino acid positions for the target and template proteins (Green): This is included in the .pir alignment. Is necessary that the number of residues in this file fit with the number of residues in the target sequence and with the number of residues in the PDB files from the templates.

- Chains for the target and template proteins (Magenta): this is included in the .pir alignment. It is used to select the chain you want to use as template for modeling from your template PDB file. In this case, our template only has one chain, so we can leave it empty.

- Name of the .pir alignment file (Orange): this is included in the modeling.py file. It tells the program the name of the input alignment between the target and the template/s.

Although the inconsistencies between file names and paths is the major source of errors while running MODELLER, the program can also fail in these situations:

- If the template sequence used in the alignment was obtained from Swissprot rather than from the PDB. Usually the sequence at the PDB is not the complete sequence. This would involve that the sequence from the structure is different from the one in the alignment. We can solve this problem by obtaining the sequence of the template running PDBtoSplitChain.

- If the template has segments with occupancy higher than 1. This means that this template has regions were more than one amino acid are being overlapped in the same position. Although this error can be solved, it is beyond the scope of this course. In such case the best option is to use a different template.

- The PDB format can have many artifacts and errors. So, in case you are having errors and you don't know why, you can always try to change the template and see if the program works. Some of these artifacts or errors can be solved if we run PDBtoSplitChain.pl on the template pdb file.

Once we have checked that our input files are correct we can run modeller. To run MODELLER execute the following command:

```
mod10.5  modeling.py
```

If everything works fine, you will obtain the next output files:

- XXXX.B9999000N.pdb → These files are the produced models, where XXXX is the input name and N is the number corresponding to the different models generated by a MODELLER run.

- XXXX.log → This file contains the information of the whole MODELLER execution. In case that there is an error, this would be reported into the .log file. Then, by looking to the .log file we should see the error and know why MODELLER has crashed.

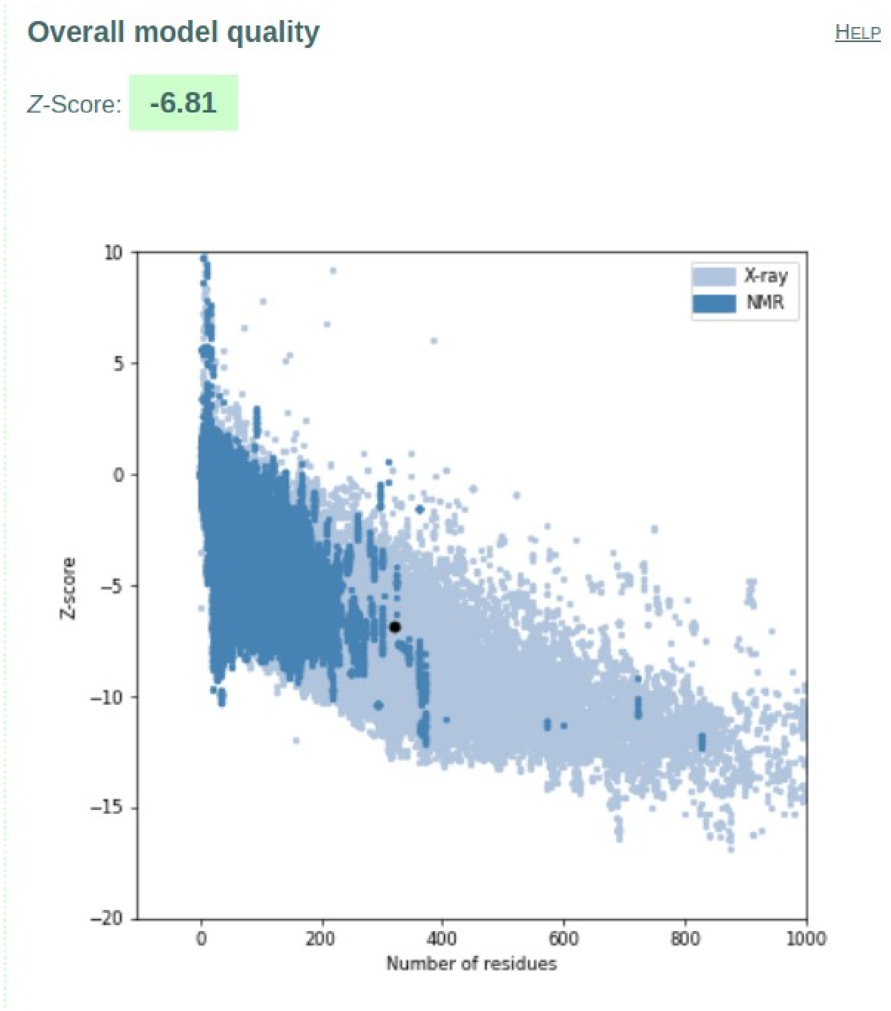
## Step 3: evaluating our models with prosa

After obtaining one protein model we should evaluate its quality to be sure that is a reliable model. We are going to do so by using statistical potentials. In particular, we will use one program called prosa that enables us to view the energy profiles of the generated models. This program is accessible through the internet at the following direction:
https://prosa.services.came.sbg.ac.at/prosa.php

Since statistical potentials are relative measurements, we should compare the prosa results for our model with the results of a reference protein. Our template is an excellent reference protein because is an experimental structure (therefore we know it is correct) and it is supposed to be similar to our model. Then, what we have to do is to run two prosa web executions: one for our model and another for our template. To do this, open two prosa web windows, in one submit your model and in the other submit your template.

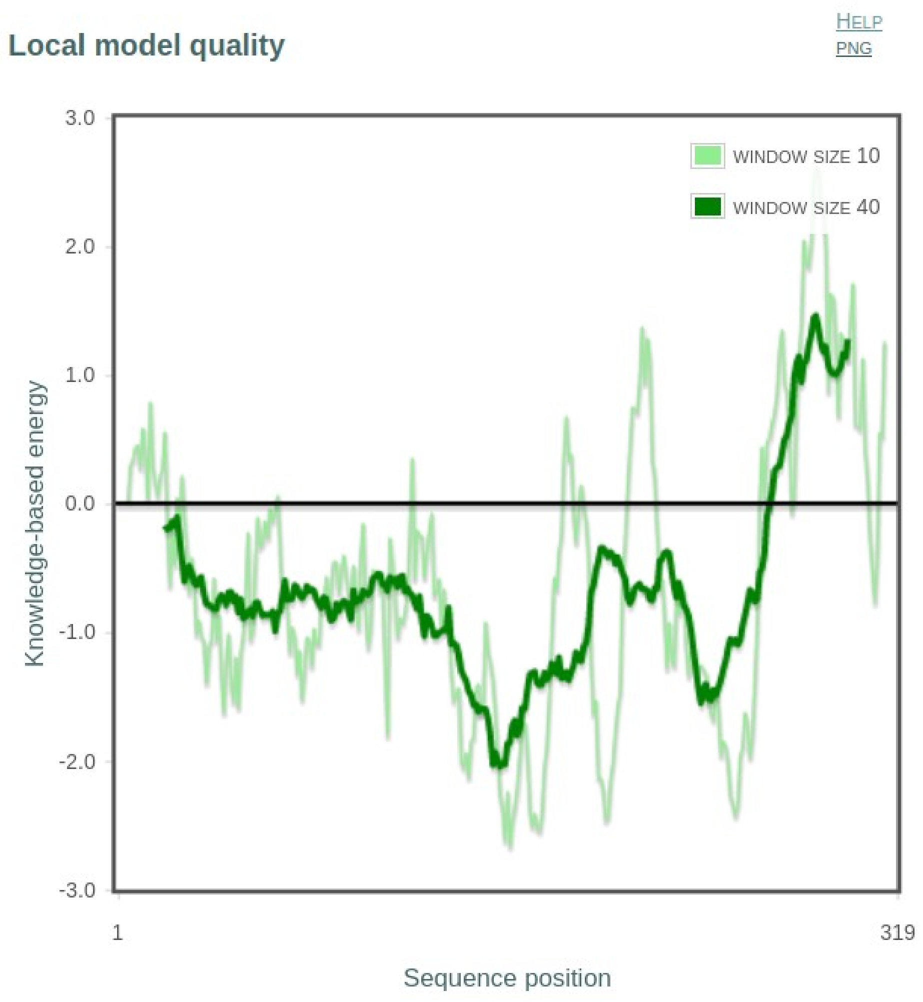
After running, prosa web will display a set of results for each analyed protein consisting of:

- A **Z-score** and a plot comparing how good is your structure in comparison with the structures in the PDB.



- An **energetic profile** showing the statistical potentials scores across the amino acid sequence. On the X axis we have the protein sequence, where each amino acid is represented by its position in the protein sequence. On the Y axis we have the energy values for our model. As you see, each amino acid has its own energy value.

  See that in this energetic profile you have two curves: one that has very extreme values and another that is more flat. This is because each curve shows the same results but using a different sliding window. When we apply a sliding window to a line plot we make that each point in the plot has the average value of all the neighboring positions. For example, with a sliding window of 20 we are making that each point in the plot has the average value for the 10 preceding and the 10 following positions in the protein sequence. This smooths irregularities and makes easier the understanding of the plot. Typically, it is recommended to work with a window size that is approximately the 10% of the sequence length. Remember that as the window is bigger we miss details in the energetic profile.



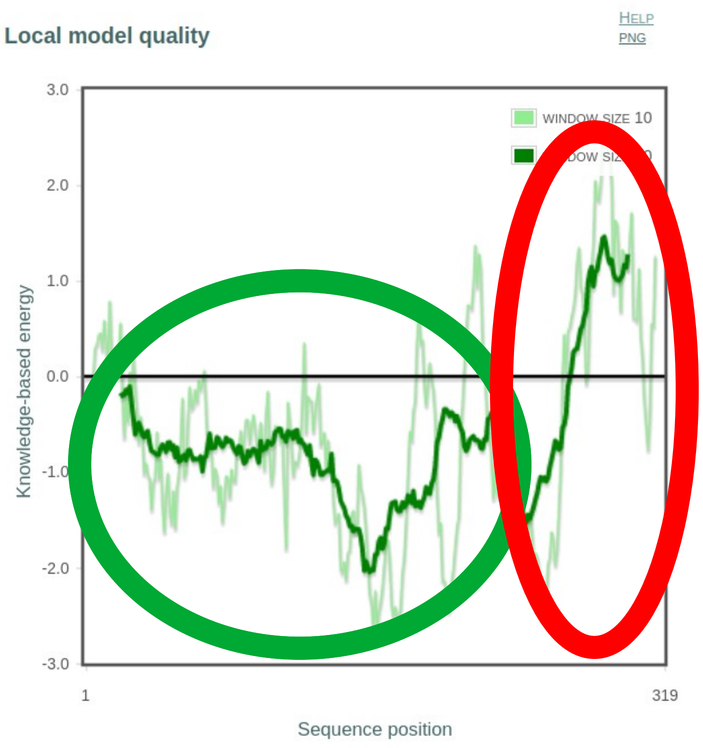
Prosa is providing three energetic values for the analysis of our protein model:

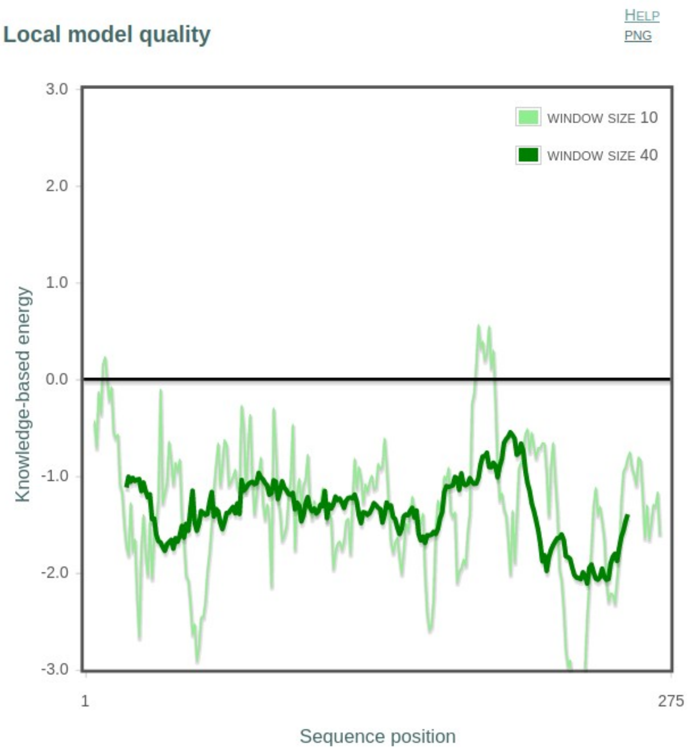
- PAIR energy: this energetic component is calculated taking into account the distances between pairs of amino acids. By default it uses distances between beta carbons. By doing so, prosa takes into account not just the location of amino acids, but also the orientation of the side chains.

- SURF energy: this energetic component is calculated taking into account the degree of exposure of amino acids. Prosa expects to find hydrophobic amino acids in the core of the protein and polar amino acids in the surface.

- COMB energy: it is a lineal combination of the PAIR and the SURF energies. Is the energy score that prosa web shows you by default.

By comparing the energy profiles we can identify that our model has some regions that are well modeled and some others that are not. We will see this by the energy scores. The lowest the energy, the most stable is the structure. For the model this is an indication that is has been well modeled. So, those regions that are showing clearly higher energies in the model in comparison with the template are likely to be wrongly modeled.





By looking at the plots (model in the left and template in the right) we identify two regions:

- A well modeled region (Green): here the energy values are similar between the model and the template.

- A wrongly modeled region (Red): here the energy values of the model are much higher than the ones of the template. These regions can be corrected, as we will see in the following practice.

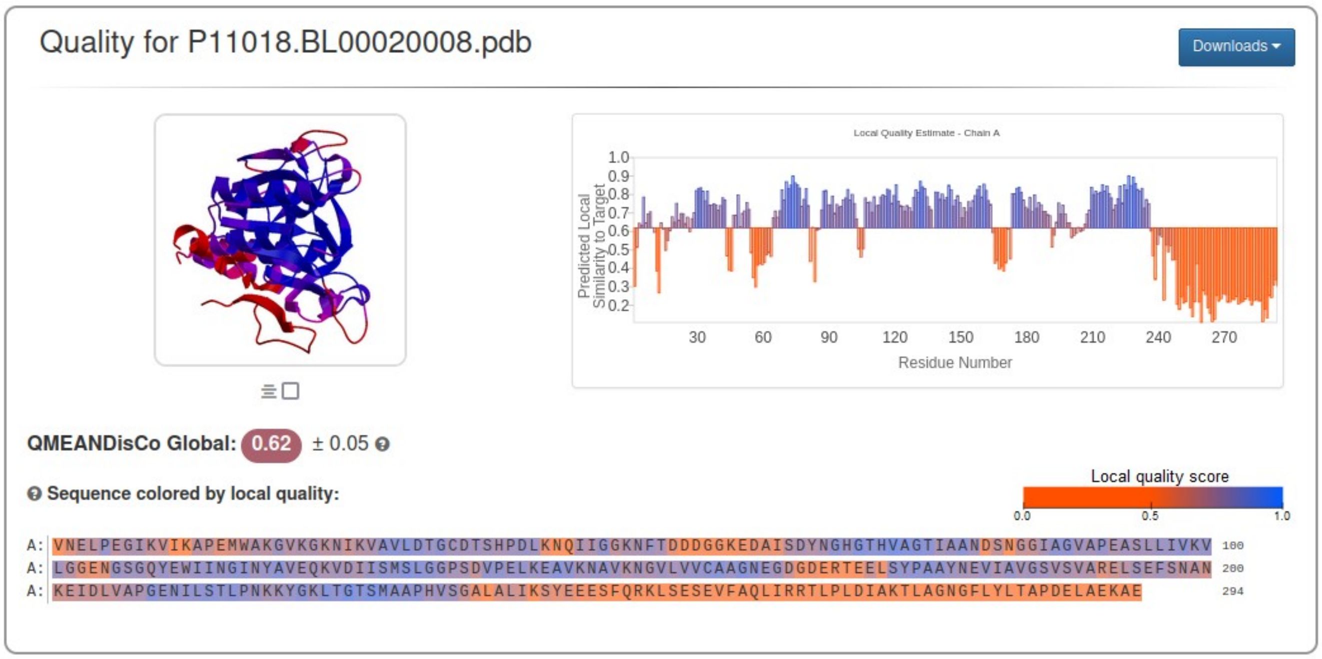
We clearly see that one part of the model is alright and the other is wrong. We have two ways of dealing with this situation:

- Removing the wrongly modeled part of the model. By doing so we will have an incomplete model, but at least we will be sure that the modeled parts are right.

- Correct the model. There are many strategies to correct the model. We will see some of them in the next practical lesson.

Another website where you can use statistical potentials to analyze protein structures is QMEAN, from the University of Basel: https://swissmodel.expasy.org/qmean/

This page takes a little longer to run (10-15 minutes) but it displays the results in a more cool way. I strongly recommend you to use this page to analyze the effect of mutations in your project. On the other hand, for the practical classes and exams, since we want to be fast, prosa web is the best option. Here you can see an example of this display:



Quality for P11018.BL00020008.pdb

**QMEANDisCo Global:** 0.62 ± 0.05

**Sequence colored by local quality:**

```
A: VNELPEGIKVIKAPEMWAKGVKGKNIKVAVLDTGCDTSHPDLKNQIIGGKNFTDDDGGKEDAISDYNGHGTHVAGTIAANDSNGGIAGVAPEASLLIVKV 100
A: LGGENGSGQYEWIINGINYAVEQKVDIISMSLGGPSDVPELKEAVKNAVKNGVLVVCAAGNEGDGDERTEELSYPAAYNEVIAVGSVSVARELSEFSNAN 200
A: KEIDLVAPGENILSTLPNKKYGKLTGTSMAAPHVSGALALIKSYEEESFQRKLSESEVFAQLIRRTLPLDIAKTLAGNGFLYLTAPDELAEKAE 294
```

You can see that both methods, prosa and QMEAN, are identifying errors in the modeling of the C-terminal of our model.

## Step 4: evaluating a set of threading models

So far, we have used prosa to compare a model with a template with the purpose of identifying wrongly modeled regions. We can also use prosa to compare different models between them. If we have produced a set of models and we want to know which model is the best, using prosa is a good strategy. Our hypothesis is that the model having the lowest energies is going to be the best modeled.

We are going to work with a set of models obtained by different threading programs. These models are inside the threading directory inside the “exercise_4” folder. You have this files inside the sitdoc cluster, so there is no need to copy them from the local computer. Then, copy the exercise_4 directory in your home directory and go to “exercise_4/THREADING/” directory.

Here we have threading models that have been obtained using different threading programs. These programs are I-tasser and phyre. You can access these programs through the internet and submit your queries. Remember how these threading programs work:

1. They contain a database of protein structures.

2. They fit the target sequence inside each one of these structures. For each fitting, energetic scores are obtained.

3. The structures showing the best scores are used to create a protein model.

Now, open a prosa web page for each of these models and compare their Z-scores and energetic profiles to know what is the best.

## **Questions from the tutorial:**

1) Obtain models for other templates that you didn't use during the tutorial. Then compare the models. Do they look the same? Do they have the same scores profiles in PROSA? Why is this happening?

2) Use the target to identify 5 new templates. Then, use these templates to build a structure-based HMM as we saw in exercise 3. Chose one of these templates and align it with the target using the structure-based HMM.

3) Search in PFAM the HMM that fits the best your target protein. Then align the template that you choose in the previous exercise with your target sequence using the HMM you just retrieved from PFAM.

4) Use the structure-based alignment and the sequence-based alignments obtained in the exercises 2 and 3. Use these alignments to make models of the target. Then compare the models obtained with different alignments. Do they look the same? Do they have the same scores profiles in PROSA? Why is this happening?

5) Try to build models for the targets contained in the "extra_targets" directory.