University Of Bergen

Faculty Of Mathematics And Natural Sciences
Department Of Informatics

INF399 - Master's Thesis in Informatics

Uncovering Amino Acids Patterns At The Membrane-Binding Interfaces Of Peripheral Proteins

Student:	Florian Sebastian Alexander Müller
Studies:	Master Of Science In Bioinformatics
Semester:	10th Semester (4th Semester In Master)
Student ID:	244545
Birth date:	24.11.1995
Phone-No.:	(+47) 99 87 25 16
E-Mail:	florianisch.flo@googlemail.com

Repository: https://git.app.uib.no/reuter-group/protrusions_msc_florian

Bergen, 16.06.2020

-- Table Of Contents --

-- Table Of Figures --

-- Table Of Tables --

-- Table Of Definitions --

Appendix

Abbreviations & Terms

Appendix 1

Amino Acids

ALA	alanine
ARG	arginine
ASN	asparagine
ASP	aspartic acid
CYS	cysteine
GLN	glutamine
GLU	glutamic acid
GLY	glycine
HIS	histidine
ILE	isoleucine
LEU	leucine
LYS	lysine
MET	methionine
PHE	phenylalanine
PRO	proline
SER	serine
THR	threonine
TRP	tryptophan
TYR	tyrosine
VAL	valine

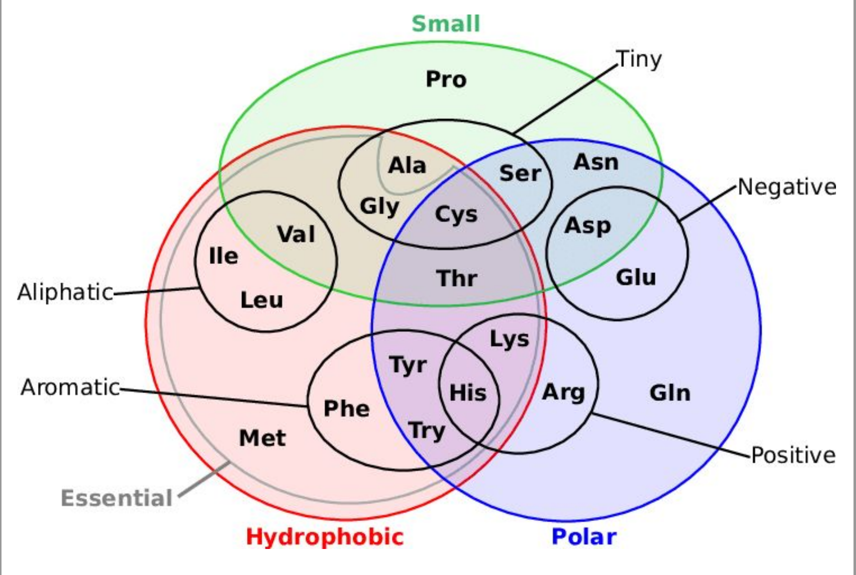


Figure appendix.1 - Venn diagram showing all amino acids and its major grouping.

This venn diagram is from Lamy et al. (2016), from the following url: https://www.researchgate.net/figure/Venn-diagram-showing-9-properties-of-the-20-amino-acids_fig2_307573998

Positively charged amino acids:
ARG, HIS, LYS

Negatively charged amino acids:
ASP, GLU

Hydrophobic amino acids:
CYS, ILE, LEU, MET, PHE, TRP, TYR

Note: According to the Wimley and White's (1996) hydrophobicity scale at membrane interfaces, the amino acid residues cysteine (CYS), isoleucine (ILE), leucine (LEU), methionine (MET), phenylalanine (PHE), tryptophan (TRP) and tyrosine (TYR) are the hydrophobic amino acids that are favourably inserted at the membrane interface. The scale is also used in the study from Fuglebakk and Reuter (2018) and I wanted to keep consistency with their study.

Chapter 1

Introduction

Chapter 1.1

Introduction

Biological membranes plays an active and essential role when it comes to the organisation of life. They do define cells and organelles, and they contain and interact with a various types of proteins which again are part of numerous pathways (Luckey, 2014; Fuglebakk and Reuter, 2018). One type of such proteins are integral proteins, such as transmembrane proteins, which are well integrated into the membrane structure. Those transmembrane proteins span through the entire membrane bilayer and function often as gateways to permit the transport of particular substances through the membrane (Luckey, 2014; Watson, Baker and Bell, 2014). Transmembrane proteins contain hydrophobic regions that are well defined, which identify their membrane interacting and embedded segments (Fuglebakk and Reuter, 2018). Another type of proteins we can find at membranes are peripheral proteins. They interact just temporarily with the membrane, bind typically via electrostatic interactions and are also able to insert hydrophobic amino acids into the membrane bilayer (Luckey, 2014). Peripheral proteins do often have a specific binding site for a lipid ligand on their membrane-binding domain (Lucky, 2014), which mainly are made of hydrophobic and positively charged amino acids (Johnson and Cornell, 1999).

Fuglebakk and Reuter (2018) stated that interfacial binding sites of peripheral proteins are poorly characterized when it comes to amino acid composition and structural patterns. To approach this problem, they defined a model of *hydrophobic protrusions* and showed that their model of *protruding, co-insertable hydrophobes* can be used to distinguish surfaces of peripheral proteins from the non-binding surfaces of transmembrane proteins. Fuglebakk and Reuter (2018) showed also that their concept of the *likely inserted hydrophobe* coincides with the binding sites of known peripheral proteins, identifying membrane-binding amino acids. In the past, the study of membrane-binding interfaces at peripheral proteins mostly focused on electrostatic energetic models and binding sites of known peripheral proteins are also usually experimentally verified (Fuglebakk and Reuter, 2018). Fuglebakk and Reuter (2018) now presented a new approach to study binding sites of peripheral proteins, considering a general and computational model of protruding hydrophobes, which easily could be used to predict potential binding sites on peripheral proteins.

In order to illustrate their protrusion definition, Fuglebakk and Reuter (2018) created some very interesting and inspiring visualizations using the VMD (Visual Molecular Dynamics) software (Humphrey, Dalke and Schulten, 1996). They show some peripheral proteins with their computed convex hull and their protrusions on a figure. But something that I really miss here at this point is a more interactive and dynamic visualization, which would make it possible to play with their protrusion definition and to visually study then the result on a peripheral protein of my interest. Since Fuglebakk and Reuters (2018) protrusion model is still quite new, there are no useful tools to visualize the model without strong programming skills. But to spread the protrusion model to the field and even to people without decent programming skills, a quick and easy to use tool is mandatory. Otherwise, the opportunity of studying peripheral proteins visually with Fuglebakk and Reuters (2018) protrusion model would become highly dependent on people's programming skills, which is not appropriate. Therefore, I specified the implementation of such a tool, which is interactive visualizing the protrusion model, as my first and major problem of my master thesis.

The study of Fuglebakk and Reuter (2018) focused mainly on a model of protruding hydrophobes, as mentioned before. However they deliberately avoided other known and import factors such as electrostatics, conformational flexibility and relative hydrophobicity in their protrusion definition, as they themselves mentioned. The explanation they give was to not overcomplicate the model and being able to isolate major components involved in membrane binding, stepwise.

Reading the study of Fuglebakk and Reuter (2018), one may get an impression that complex, complementary electrostatic models are overrepresented in the study of peripheral protein membrane binding. They stated that the importance of hydrophobes at binding sites may have been underrated, which is why they developed their protrusion model. But they also stated that considering electrostatic properties may improve their model. We know that complementary electrostatic interactions between peripheral proteins and the membrane play an important role. We do find anionic lipids in the membrane (Lucky, 2014) and we do find mostly hydrophobic and positively charged amino acids at known membrane-binding domains on peripheral proteins (Johnson and Cornell, 1999).

I think that a solid computational model for binding site prediction on peripheral proteins should also consider electrostatic properties, based on the biological knowledge we have today. In order to investigate electrostatic properties at binding sites of peripheral proteins in a way that keeps consistency with Fuglebakk and Reuters (2018) previous study, we absolutely should have a look at the amino acids we can find around hydrophobic protrusions, if they are charged or not. I specified this to be my second, major problem of my master thesis. I mainly focused on characterizing the environment of hydrophobic protrusions in peripheral proteins and compared that to the reference dataset, using the same datasets as Fuglebakk and Reuter (2018) for that purpose.

I started to approach my problems with a reproduction of Fuglebakk and Reuters (2018) findings, using my own implementation. Reproducing the results assure correctness of my understanding about the definition of protrusions in general and gives me a good start on the project. Next, I implemented a plugin for the popular molecular viewer UCSF Chimera (Pettersen *et al.*, 2004), which computes and shows convex hulls and protrusions. The plugin makes the visual study of the protrusion model on a peripheral protein of interest very simple and interactive for everyone. The choice of using a popular and freely available software as UCSF Chimera was important. Finally, I extended the work of Fuglebakk and Reuter (2018), focusing on the neighborhood of hydrophobic protrusions and proposed a series of analysis and definitions to characterize the environment. I introduced some new definitions such as *the neighbourhood, the neighbourhoods hydrophobicity* and *the neighbourhoods charge*. Later on I also defined the *exposed neighbourhood*, which excludes amino acids that may be positioned too deep into the protein structure and unlikely to be membrane-binding. I then applied my definitions to the datasets from Fuglebakk and Reuter (2018). The results show that all twenty amino acids are present in the neighbourhood of hydrophobic protrusions. A specific pattern or model at the mean amino acids composition is not found. However, we can see that the combination of hydrophobic and positively charged amino acids in a neighbourhood of a hydrophobic protrusion from the peripheral protein dataset becomes dominating compared to neighbourhoods from the reference dataset. An approach like this is also suitable, because I could maintain consistency with the study from Fuglebakk and Reuter (2018).

Chapter 2

Background & State Of The Art

Chapter 2.1

Biology

As proteins, mainly peripheral proteins, play a major role in my master thesis, I do like to introduce the biological part with the central dogma of biology which is described by Watson, Baker and Bell (2014). The central dogma describes the three main levels of cell mechanisms that are involved to build proteins.

A gene on the deoxyribonucleic acid (DNA) becomes transcripted to messenger ribonucleic acids (mRNA) and the resulting mRNA strands become translated to proteins. The study of structures and functions on these three levels in the central dogma are often described using the omics neologism, respectively *genomics* (DNA / Gene level)*, transcriptomics* (RNA / Transcript level) and *proteomics* (protein level)*.* It is important to understand on which level this thesis focused on, the proteomics, in order to understand the correct level of biological abstraction and where we are in dogma in general.

It is the ribosome protein complex that performs the translation from a mRNA strand to a protein. Here, the ribosome protein complex reads the mRNA strand and matches transfer ribonucleic acids (tRNA) to mRNA nucleotide triplets (codons). The amino acid attached to the tRNA binds to the peptide chain, then the tRNA releases again. If the translations process finished, the polypeptide chain releases from the ribosome complex and is ready to be matured by post-traductional complexes in the cytoplasm or by other cells organelles like the golgi apparatus or the endoplasmic reticulum (Watson, Baker and Bell, 2014).

As stated by Watson, Baker and Bell (2014), the structure of a protein can be described in four levels. The sequence of amino acids in the polypeptide chain is simply the primary structure. We can observe more or less local conformations of the chain to more complex three-dimensional structures, called the secondary structure. Typical secondary structure elements are alpha-helices and beta-sheets. The organisation of multiple alpha-helices and / or beta-sheets form the tertiary structure where the secondary structure elements are linked by *e.g.* loops and turns between secondary structures. Many proteins are also made of more than just one polypeptide chain. In these structures, the organisation of multiple tertiary structures forms a larger protein complex, which is called the quaternary structure of the protein.

2.1.1 Biological membranes, peripheral proteins and transmembrane proteins

Some types of proteins are able to bind to biological membranes (Luckey, 2014; Watson, Baker and Bell, 2014). The data of this master thesis covers two types of proteins that are able to bind to the membrane, peripheral proteins and transmembrane proteins.

Biological membranes consist primarily of two layers of lipids and other components *e.g.* different types of proteins, carbohydrates and cholesterol, as shown on the figure 2.1.1. This main membrane model is described as the fluid mosaic model by Singer and Nicolson (Luckey, 2014). Luckey (2014) describes that a lipid is made up of a hydrophilic head and a hydrophobic tail. Because of the hydrophobic characters of their tail, lipids spontaneously form bilayers. Their hydrophilic heads align and orient towards the aqueous environment and their tails are shielded from the solvent inside. The lipids are actually forming a hydrophobic barrier.

There exists different types of lipids, which vary by their tail length, the number of double bonds, their saturation and by their headgroups. Some lipids can be anionic lipids, which means that their head is negatively charged.

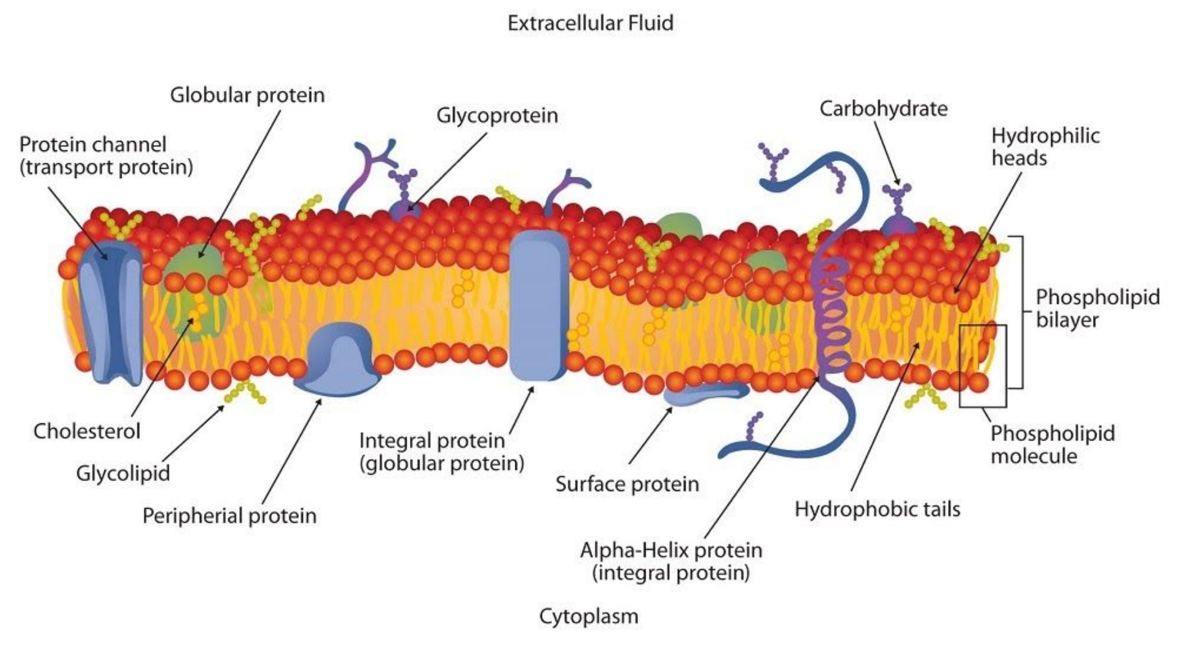


Figure 2.1.1 - The lipid bilayer and typical proteins in and around the bilayer.

This figure is from the following URL (Libretexts, 2013):
https://chem.libretexts.org/Bookshelves/Biological_Chemistry/Supplemental_Modules_(Biological_Chemistry)/Lipids/Applications_of_Lipids/Lipid_Bilayer_Membranes

Transmembrane proteins, as the naming already indicates, are a type of integral proteins that are able to live in the membrane and they span through the entire bilayer (Luckey, 2014). Those proteins can *e.g.* be anion channels and are highly integrated into the membrane structure, as stated by Luckey (2014). Furthermore, those transmembrane proteins consist of a hydrophobic part within the bilayer and one or more hydrophilic parts, outside of the membrane, either at one or both sites of the membranes bilayer.

However, some proteins at the membrane can bind to and release from the membrane more easily than integral proteins (Luckey, 2014). They are called extrinsic proteins in the fluid mosaic model. Peripheral proteins are such types of proteins and they can bind the lipid headgroups surface at the surface of the membrane. Peripheral proteins can also insert one or a few amino acids into the hydrophobic part of the membrane. In contrast to transmembrane proteins, peripheral proteins are able to enter the membrane without damaging its structure. Peripheral proteins typically bind via electrostatic interactions to the membrane, but they often also insert hydrophobic amino acid residues into the membrane (Luckey, 2014).

As a consequence so are known binding sites mainly made of hydrophobic and positively charged amino acids (Johnson and Cornell, 1999). There are some exceptions to the rule *e.g.* some proteins do have very few positively charged amino acids. Another example is the role of aromatic amino acids in bacterial phospholipases. They do not necessarily insert into the hydrophobic part of the lipid bilayer. However they interact with choline lipids and engage in very specific interactions (Grauffel *et al.*, 2013). In general, typical membrane-binding domains such as C1, C2, FYVE and PH have a specific binding site that is usually for lipid ligand. This site is often supported by additional, less specific binding sites and these differ between different proteins (Luckey, 2014).

The description of membrane binding sites as a collection of hydrophobic and positively charged amino acid residues is old and needs to be updated, heaving the complexity of protein-lipid interfaces uncovered the last five to ten years in mind. For these reasons, this study aims at mapping patterns in amino acid distributions at the membrane binding interface of peripheral proteins. From a biological point of view, looking for hydrophobicity and electrostatic properties in the peripherals protein structure makes sense in order to find potential membrane binding sites, based on the biological knowledge we have today.

Chapter 2.2

A model for hydrophobic protrusions on peripheral membrane proteins, Fuglebakk and Reuter (2018)

The main reference for this study was the study from Fuglebakk and Reuter (2018) with the title “A model for hydrophobic protrusions on peripheral membrane proteins”. They defined a general model of *hydrophobic protrusions* and showed that their model of protruding, co-insertable hydrophobes can be used to distinguish surfaces of peripheral membrane proteins from the surfaces of transmembrane proteins in their reference dataset. Fuglebakk and Reuter (2018) showed also that their concept of the *likely inserted hydrophobe* can be used to identify potential membrane-binding residues of peripheral proteins, as the likely inserted hydrophobe coincides with the binding sites of known peripheral proteins.

In this section, I give an overview about Fuglebakk and Reuters (2018) most important key definitions and concepts. Note that I used the same datasets in my study as Fuglebakk and Reuter (2018), so the details about the datasets used are described in chapter 3.1.

Convex hull
Fuglebakk and Reuter (2018) computed a convex hull for a given protein. The convex hull of a set of points is the smallest convex polygon that encloses all points. The convex hull is computed on the set of all alpha carbon (Cα) atoms and beta carbon (Cβ) atoms of the protein. Their convex hull is computed with the Scipy’s (Millman, Jarrod Millman and Aivazis, 2011) implementation of the Qhull (Barber *et al.*, 1996) algorithm.

Local protein density
Within a distance of ten angstroms (10 Å) from a beta carbon (Cβ) atom of interest, the local protein density is defined as the number of all neighbour alpha carbon (Cα) atoms and beta carbon (Cβ) atoms.

Low local protein density
With the low local protein density, Fuglebakk and Reuter (2018) described a local protein density that is less than twenty-two.

Protrusion
A protrusion describes amino acids that are protruding to the outside with respect to the rest of the protein. In order to become a protrusion, the amino acid's beta carbon (Cβ) atom needs to be a vertex of the computed convex hull and needs to have a low local protein density.

Hydrophobic protrusion
When a hydrophobic amino acid becomes a protrusion, then it simply is a hydrophobic protrusion. Sometimes, Fuglebakk and Reuter (2018) also call such amino acids as *the protruding hydrophobes*. They used the Wimley and Whites (1996) hydrophobicity scale at membrane interfaces as a reference in order to define which amino acids are treated as hydrophobic. As already stated in appendix 1, the hydrophobic amino acids are cysteine (CYS), isoleucine (ILE), leucine (LEU), methionine (MET), phenylalanine (PHE), tryptophan (TRP) and tyrosine (TYR). Those are favourably inserted at the membrane interface.

Co-insertable
A co-insertable is a hydrophobic protrusion that does have other hydrophobic protrusions in its close neighbourhood. Here, the close neighbourhood means that two hydrophobic protrusions need to be a pair on the same convex hull edge in order to become co-insertables.

The likely inserted hydrophobe
Fuglebakk and Reuter (2018) defined the likely inserted hydrophobe as the protruding hydrophobe (hydrophobic protrusion) with the highest number of co-insertables and the lowest local protein density. A protein just has one likely inserted hydrophobe, unless the protein does not have any hydrophobic protrusions at all. If multiple protruding hydrophobes fulfill equality at both the highest number of co-insertables and the lowest local protein density, then the final likely inserted hydrophobe is chosen at random.

Fuglebakk and Reuter (2018) showed clearly that hydrophobic protrusions are more common in peripheral proteins than in their reference dataset which consists of non-membrane binding protein surfaces. They also showed that protrusions in low density regions of peripheral proteins are more often hydrophobic amino acids compared to their reference data. The co-insertable feature distinguishes between peripheral proteins and the reference dataset quite well which means that hydrophobic protrusions on peripheral proteins tend to be more co-insertable than hydrophobic protrusions from the reference dataset. In addition Fuglebakk and Reuter (2018) showed that hydrophobic protrusions are more frequent on turns, bends and alpha helices in the peripheral protein dataset compared to the reference dataset. Finally they showed that the likely inserted hydrophobe coincides with the binding sites of known peripheral proteins and that large aliphatic and aromatic amino acids are overrepresented protrusions on peripheral proteins.

The study of membrane-binding peripheral proteins mostly focused on electrostatic energetics and models in the past (Fuglebakk and Reuter, 2018). Based on Fuglebakk and

Reuter (2018) however, we are now able to approach the study of membrane-binding peripheral proteins considering a hydrophobic model. Their new approach may speed up the study of potential, membrane-binding peripheral proteins in the future.

The following figure 2.1.2 illustrates the protrusion model, showing an example structure, its computed convex hull, protrusions and hydrophobic protrusions.

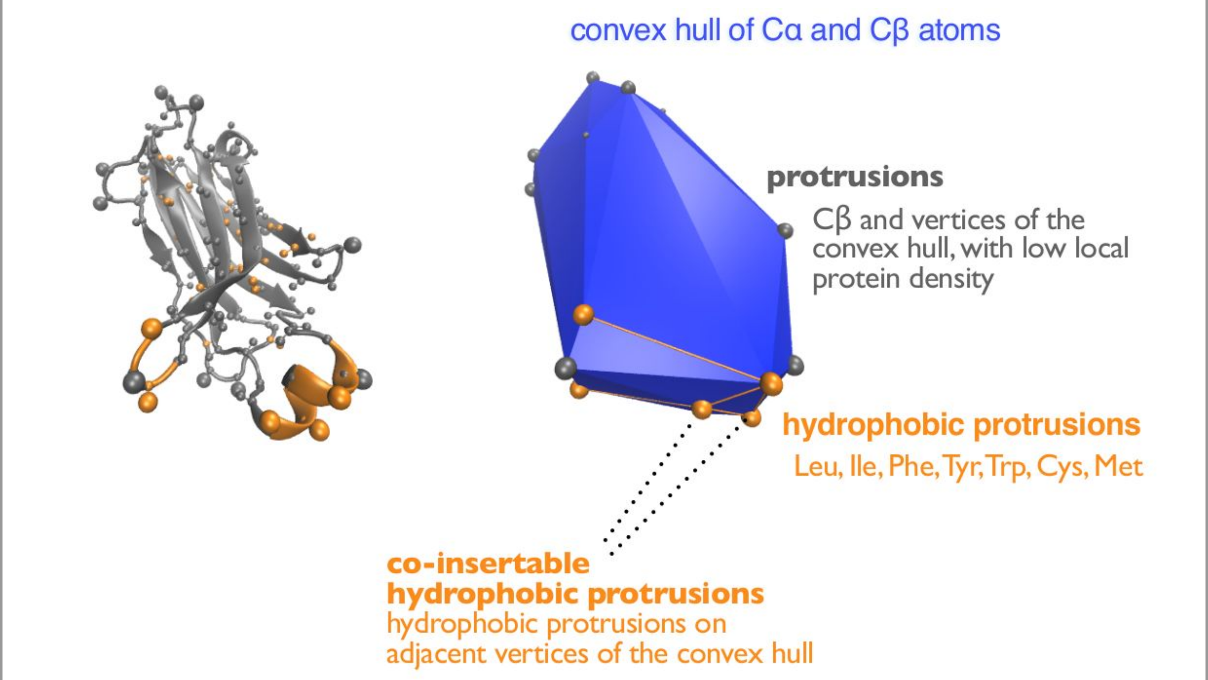

convex hull of Cα and Cβ atoms
protrusions
Cβ and vertices of the convex hull, with low local protein density
hydrophobic protrusions
Leu, Ile, Phe, Tyr, Trp, Cys, Met
co-insertable hydrophobic protrusions
hydrophobic protrusions on adjacent vertices of the convex hull

Figure 2.1.2 - The model for hydrophobic protrusions.

This figure shows a representation of the C2 domain of human phospholipase A2, with the pdb ID 1RLW. The structure to the left on the figure shows the protein structure. You can see the computed convex hull of the structure to right on the figure. The protrusion model is applied to the protein and protrusions are highlighted as large, grey balls. Hydrophobes are highlighted orange. The figure is from Fuglebakk and Reuter (2018).

Chapter 3

Materials & Methods

Chapter 3.1

Datasets

3.1.1 What data is used & where does the data come from?

I used the datasets from Fuglebakk and Reuter (2018) for this master thesis. The data is separated in two main datasets, the peripheral protein dataset and the reference dataset. Both datasets come as comma-separated CSV files, containing PDB IDs that refer to proteins from the Orientation Of Proteins In Membrane (OPM) database (Lomize *et al.*, 2012; Fuglebakk and Reuter, 2018).

Fuglebakk and Reuter (2018) did not provide any protein data bank (PDB) files with actual protein data but only the PDB IDs, so in theory I need to download the data from the OPM database myself based on the PDB IDs they provided. This can be done by a simple download script. Fortunately, the Reuter Group (*Reuter Group – CBU*, no date) where I am writing this master thesis had already a file server providing all necessary PDB files. In order to obtain my local copy of the data, I just needed to go over some folders and do a copy & paste of relevant PDB files.

The peripheral protein dataset (S1 Dataset) contains 1012 peripheral proteins and all peripheral proteins in this dataset do have the OPM classification “Monotopic/peripheral”. Fuglebakk and Reuter (2018) have drawn attention to the possibility that the dataset may contain false positives, because the membrane binding is not asserted by experiment for all peripheral proteins. They also stated that OPM has strict criteria when it comes to include peripheral proteins into the database, so the possibility of false positives is considered as quite low.

The “non-binding surfaces” dataset (S2 Dataset) contains 495 transmembrane proteins from the OPM database. As the naming already implies, this dataset consists of the solvent exposed regions of transmembrane proteins. These are regions that do not interact with the membrane somehow and this can be asserted with a high level of confidence. Fuglebakk and Reuter (2018) defined such regions as all amino acids whose alpha carbon (Cα) atom coordinates are at least fifteen angstrom (15 Å) away from the hydrocarbon region of the membrane model. As these data are retrieved from OPM, the PDB file contain information about the position of the membrane, notably the ZHDC parameter in the OPM model which describes the hydrocarbon region (*OPM*, no date; Lomize *et al.*, 2012). When computing the proteins convex hull, the whole transmembrane protein is considered. All other computations are restricted to the solvent exposed regions (Fuglebakk and Reuter, 2018). For simplicity, I will call this dataset the “reference dataset”.

3.1.2 The PDB file format & ATOM records

As mentioned in the previous subchapter 3.1.1, the protein data comes as protein data bank files, PDB files for short. The PDB file format (Berman *et al.*, 2000) has roots back to the 1970s. PDB files are text files and one PDB file is representing a whole protein complex of interest. It can even contain multiple structures. I summarize its main features below.

The protein structure is stored in many records, which are organized in sections (*Atomic Coordinate Entry Format Version 3.3*, no date). To get a general impression about the file structure and how complex PDB files can be, the following list below shows what sections and records can be found in common PDB files. A PDB file may not contain all sections or all records. Therefore, PDB files from different databases can differ a lot.

- **Title section**, a general section that contains key information about the pdb file, experiments and structures. Records you find here are HEADER, SOURCE, AUTHOR, OBSLTE, KEYWDS, REVDAT, TITLE, EXPDTA, SPRSDE, SPLT, NUMMDL, JRNL, CAVEAT, MDLTYO, REMARKS and COMPND.

- **Primary structure section**, which contains sequence information. Records you find here are DBREF, DBREF1/DBREF2, SEQADV, SEQRES and MODRES.

- **Heterogen section**, which contains descriptions for all non-standard residues. Records you find here are HET, HETNAM, HETSYN and FORMUL.

- **Secondary structure section**, which describes secondary protein structures like helices and sheets. Records you can find here are simply HELIX and SHEET.

- **Connectivity annotation section**, describes disulfide bonds or other linkages. Records you can find here are SSBOND, LINK and CISPEP.

- **Miscellaneous feature section**, which describes other molecule properties. Such properties does often not have a default record type and are also described in the title section with REMARKS records. The only default record here is the SITE record.

- **Crystallographic & coordinate transformation section**, which describes the resulting geometry of crystallographic experiments and coordinate system transformations. Records you can find here are CRST1, ORIGXn, SCALEn, MTRIXn.

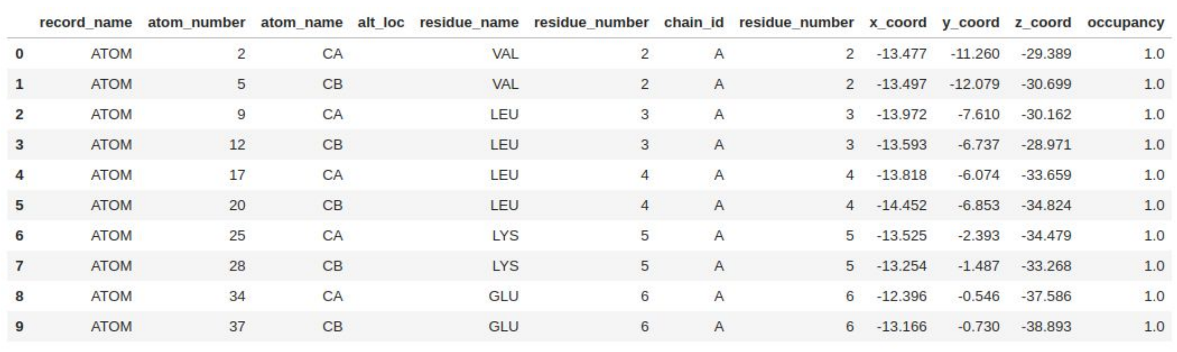
- **Coordinate section**, which contains all atomic coordinates and types. Records you can find here are MODEL, ATOM, ANISOU, TER, HETATM and ENDMDL.

- **Connectivity section**, which specifies connectivity and relates the coordinate section to the connectivity annotation section. The record you can find here is simply called CONECT.

- **Bookkeeping section**, that contains information about the file *e.g.* if the file contains multiple structures, how much lines some types of records may have for better parsing and a mark for the end of the PDB file. Records here are MASTER and END.

The title section is interesting in the sense that you can get general information about the structure *e.g.* which technique was used to resolve the structure. The most interesting section for us is the coordinate section. It contains ATOM records, that describe all atoms for a given protein structure. An ATOM record contains the following columns.

- "ATOM", the record name
- atom serial number
- atom name
- alternative location indicator
- residue name
- chain identifier
- residue sequence number
- iCode - code for insertion of residues
- x,y,z coordinates in angstroms (Å)
- occupancy
- temperature factor
- element symbol
- charge on the atom



	record_name	atom_number	atom_name	alt_loc	residue_name	residue_number	chain_id	residue_number	x_coord	y_coord	z_coord	occupancy
0	ATOM	2	CA		VAL	2	A	2	-13.477	-11.260	-29.389	1.0
1	ATOM	5	CB		VAL	2	A	2	-13.497	-12.079	-30.699	1.0
2	ATOM	9	CA		LEU	3	A	3	-13.972	-7.610	-30.162	1.0
3	ATOM	12	CB		LEU	3	A	3	-13.593	-6.737	-28.971	1.0
4	ATOM	17	CA		LEU	4	A	4	-13.818	-6.074	-33.659	1.0
5	ATOM	20	CB		LEU	4	A	4	-14.452	-6.853	-34.824	1.0
6	ATOM	25	CA		LYS	5	A	5	-13.525	-2.393	-34.479	1.0
7	ATOM	28	CB		LYS	5	A	5	-13.254	-1.487	-33.268	1.0
8	ATOM	34	CA		GLU	6	A	6	-12.396	-0.546	-37.586	1.0
9	ATOM	37	CB		GLU	6	A	6	-13.166	-0.730	-38.893	1.0

Figure 3.1.1 - Example of an ATOM record table.
This figure shows the head (first ten rows) of all alpha carbon (CA) and beta carbon (CB) atoms from the 1kcm peripheral protein structures ATOM record table. This structure does not have any alternative locations (alt_loc). Not all columns from the ATOM record table are shown, but the columns shown are the most relevant for this project.

At ATOM records, the most interesting information for this project are the atom serial number & name, residue sequence number & name, the chain identifier and the x,y,z coordinates. Within a structure, we can use those particular columns to identify the atom on the structure and we know to which amino acid residue the atom belongs to. The coordinates are used in convex hull and distance computations.

PDB files from protein databases are obtained with methods such as X-ray crystallography, NMR spectroscopy and cryo-electron microscopy (*PDB101: Learn: Guide to Understanding PDB Data: Methods for Determining Structure*, no date; Berman *et al.*, 2000). A protein structure can not be completely static when it becomes measured. An atom can be very agile and found to be on different locations as it moves while measuring. Here the occupancy column comes in. If the occupancy is one it means that in hundred percent of the cases the atom of interest is observed at its position. If an atom is observed at multiple locations, it will results in different ATOM records, a different mark on the alternative location and a different occupancy value. It is important to remember this when we read PDB files, because we may need to filter for the first alternative location with the highest occupancy, the most observed position.

As mentioned before, PDB files can contain many structures. This is not the case for the given data from Fuglebakk and Reuter (2018). Here, one PDB is representing just one structure. Sometimes we find alternative locations for some atoms, but it is still representing the same protein. Investigating the data from Fuglebakk and Reuter (2018), the x-ray crystallography can lead to multiple, alternative locations often.

For the reference data, the HETATM record from the coordinate section is also of interest. It contains dummy atoms marked with N and O and with the residue name DUM. These dummy atoms represent the membrane boundary planes as described by OPM (*OPM*, no date).

Chapter 3.2

Software

3.2.1 Programming language & environment

I used Python 2.7.17 (Van Rossum, G. & Drake Jr, F.L., 1995) for this master project. Python is a very common and popular programming language for data scientists. The popularity is mainly based on its simplicity compared to other programming languages. Many open source packages are available and those are often well documented. Python is also popular because of its quite big community. You can search the internet for almost every problem you may encounter and find possible solutions and examples written in Python.

It was important to me to choose a programming language and go with it for the whole project. I know that Python 2.7 is outdated soon and that Python 3 becomes the new Python standard. But I have chosen Python 2.7 for this project anyway, because I wanted to develop a plugin for UCSF Chimera (Pettersen *et al.*, 2004) to visualize protein convex hulls and protrusions. The popular molecular viewer UCSF Chimera is written in Python 2.7 and this was a strong argument for using version 2.7. At this point it is worth mentioning that parts of Fuglebakk and Reuters (2018) code also is written in Python 2.7. In other words, using Python 2.7 maximizes the compatibility to the software I will use and to prior work on this project.

I considered other programming languages of course. My first and main programming language is Java. It presents many advantages for application development and web development. However a disadvantage with Java (Arnold, Gosling and Holmes, 2012) is that one needs to compile the code prior to running it. There are no interactive scripting possibilities like in *e.g.* R or Python. There are possibilities to trick this with smart compilations and real time use of parts of one’s application, but it would not be efficient for this project. One can achieve a lot with Java, but interactive shell scripting and data science is not efficient because of the nature of Java.

R scripting (*R: The R Project for Statistical Computing*, no date) presents a lot of advantages and a lot of useful features such as the interactive scripting console that R offers data scientists and statisticians. Together with R-Studio (*RStudio | Open source & professional software for data science teams*, no date) the R programming language is a very suitable candidate for statistics and data analysis. A disadvantage with R is that relevant R packages often are outdated or badly documented, especially bioinformatic packages. Those packages are often difficult to install and do have a lot of dependencies. I also noticed that R does perform poorly when it comes to loops and complex implementations. One needs to vectorize the implementation of algorithms and to use the built-in functions to archive high performance. Otherwise, one needs to implement an R package in C++, using *e.g.* the RCPP package (*Rcpp · Advanced R*, no date). This makes the implementation of complex computations difficult again, as we would need to operate with multiple programming languages to keep a decent runtime.

Interestingly Python does have a better performance than R (Kan, 2018). Kan's article on *Towards Data Science* (Kan, 2018) claims that the hype of R is over and that Python is one of the most popular programming languages among data scientists these days. Moreover, I wanted to learn a new programming language. Altogether Python was a natural choice for my master thesis' project.

My final setup is based on the package manager Anaconda 4.8.2 (*The World's Most Popular Data Science Platform | Anaconda*, no date). I used Anaconda to set up multiple python environments for programming. By doing so I can avoid dependency problems of packages while programming. I used *e.g.* one environment for my UCSF Chimera plugin development and another environment for my data analysis.

For the data analysis, I used Jupyter Lab 0.33.12 (*Project Jupyter*, no date). Jupyter Lab is a perfect tool that combines notes in plain text or markdown with interactive code blocks in a jupyter notebook file. It is perfect to write notes while doing analysis with python scripts and that is why I used this software. As mentioned before, I used UCSF Chimera (Pettersen *et al.*, 2004) as the main molecular viewer in this project. To program the UCSF Chimera plugin, I used the PyCharm Community IDE ('PyCharm', no date).

You can find all my code, all my jupyter notebooks and results in the projects repository. The version control of this repository is Git (*Git*, no date). This is the standard for version control with the universities GitLab server.

The repository is located at the following link:
https://git.app.uib.no/reuter-group/protrusions_msc_florian

3.2.2 Python packages and additional packages / software

In this section, I am giving an overview of all python packages I used. I provide a short description of what each package is and what I used it for. Every package is downloaded with the anaconda package manager (*The World's Most Popular Data Science Platform | Anaconda*, no date).

Before I start listing all the packages, I do like to mention that I ran into some package problems when I implemented my plugin for UCSF Chimera (Pettersen *et al.*, 2004). I was not able to add the Scipy package to the UCSF Chimera software, so I needed alternatives for the computations of both the convex hull and the distances. I implemented a distance algorithm myself with Numpy but I decided to not implement Qhull (Barber *et al.*, 1996) on my own. For the Qhull implementation I used Jinxuan Wus implementation that I found on Github (jinxuan, no date). You can read more about my plugin in the chapter 4.2.

Numpy 1.16.5 (Oliphant, 2015) is a fundamental math library for scientific computing. It is a dependency for other packages, *e.g.* Pandas and SciPy. Numpy is my primary math and array library in this project. I mostly used it to find unique values in a list or for some powerful linear algebra operations with Numpy functions and arrays.

Pandas 0.24.2 (McKinney, 2010) is a fast and powerful library for data analysis. I mostly used Pandas to organize data in tables. Pandas provides a very fast query function that makes it possible to find data in tables with more than millions of rows within less than a second. Pandas is also based on Numpy and a table can easily be converted to a Numpy array with the to_numpy() function. Jupyter Lab (*Project Jupyter*, no date) can view Pandas tables in a good, visual way which is important when you want to look at some values in the data.

Biopandas 0.2.4 (Raschka, 2017) is a package that helps to import and work with biology data such as proteins, and the pdb file format in general. It converts any import to a pandas data table, ready to use for data scientists. It also provides some better export functionality and fetching from biology databases. I mainly used Biopandas to import pdb files directly into a Pandas table.

Matplotlib 2.2.3 (Hunter, 2007) is a library for plotting and that is what I used this package for. Most of my plots / figures are based on Matplotlib and got modified by this package. I also looked into ggplot2 as an alternative to Matplotlib (as I know ggplot2 from R), but I decided to stick with Matplotlib as this library is more commonly used and important for some other packages I used.

Seaborn 0.9.0 (Oberoi and Chauhan, 2019) is a data visualization library. It is based on Matplotlib and draws beautiful plots for any Pandas organized data of interest. I used Seaborn for my main plots / figures.

Scipy 1.2.1 (Millman, Jarrod Millman and Aivazis, 2011) is a data science library with a lot of mathematical, scientific, statistical and engineering algorithms. The core packages SciPy depends on are Numpy, Pandas, Matplotlib and some more. I mainly used Scipy because of its Qhull (Barber *et al.*, 1996) algorithm that I used for protein convex hull computations. I also used Scipy for efficient and fast distance computations.

TQDM 4.36.1 (Costa-Luis and da Costa-Luis, 2019; Oberoi and Chauhan, 2019) is not just a progress bar, but it is the progress bar. In this master project, I really began loving TQDM. Almost every loop is created with TQDM, showing me always some statistics on how my loops performed. While running code, TQDM shows the iterations per seconds in realtime and predicts the remaining time needed to finish all iterations. Finally it shows how long it took. In most of the cases, it was just a visual gimmick, but sometimes it actually was very useful to get an idea about how good or bad my implementation was. It helped me to improve very slow parts.

FreeSasa 2.0.2 (Mitternacht, 2016), a python module of FreeSasa, is used to compute accessible surface areas (SASA) in this project.

MMTK 2.9.3 (Hinsen, 2000), the Molecular Modeling ToolKit (MMTK) was used by Fuglebakk & Reuter (2018) to import pdb files in their code. I needed to install this package (and every dependency) in a separate anaconda environment. I mainly used this package to investigate differences when I reproduced Fuglebakk & Reuters (2018) results, see chapter 4.1. I used Biopandas in my setup instead, as Biopandas is more modern and works better together with all my other packages (Pandas, Scipy, Seaborn, etc).

Chapter 3.3

Methods - Approaching the problem

This section gives you an overview about how I did approach the problem and the computational strategy I have set up. Shortly summarized, I started by working on reproducing Fuglebakk and Reuters (2018) computations with my own implementation. Then, I implemented a convex hull plugin for UCSF Chimera, which also can show protrusions. Finally, I extended the work of Fuglebakk and Reuter (2018) by conducting new analyses of peripheral proteins and the reference dataset. In particular I focused on the neighborhood of protrusions and proposed a series of analysis and definitions to characterize the environment of the protrusions in peripheral membrane proteins.

Step 1
Because Fuglebakk and Reuter (2018) used the outdated MMTK package (Hinsen, 2000) in their implementation, I had to write my own code for the computation of protrusions and hydrophobic protrusions. I decided to use packages that are more common today and well maintained, so that my code is sustainable. Examples are Biopandas (Raschka, 2017) and Pandas (McKinney, 2010), which can be used with the newer Python 3. It was important that my implementation was able to reproduce Fuglebakk and Reuters (2018) results and this is presented in details in chapter 4.1. Implementing their definition of protrusions and hydrophobic protrusions on my own was also very useful to maximize my understanding about these definitions and their paper in general.

The pseudocode on the next page (figure 3.3.1) shows the main steps of my implementation. The coordinates of all alpha carbon atoms (Cα) and beta carbon atoms (Cβ) were extracted from the structures ATOM record table. Based on those coordinates, the structure's convex hull and a distance table were computed. Finally the algorithm iterates over all Cα and Cβ atoms from the table and checks if a particular atom fulfills the conditions to be defined as a protrusion. If an atom fulfills all criterias to be a protrusion or hydrophobic protrusion, then it becomes added to the corresponding list, which will be returned by the end of the function.

My final implementation differs from this simple pseudocode of course. I added native arrays to be able to do some caching for parts of the ATOM record table. This gives faster loop computations as values just will be looked up by an index instead of querying them every iteration from the pandas table. I added some code lines to find co-insertables too. I also checked if the amino acid residues Cα atom coordinates are at least fifteen angstrom (15 Å) away from the hydrocarbon region of the membrane model for all proteins from the reference dataset.

```
function findProtrusions(table){
    distance = 10; density = 22;
    coordinates = table[["x_coord", "y_coord", "z_coord"]];
    convex_hull = computeConvexHull(coordinates);
    distances = computeDistances(coordinates);

    protrusions = [];
    hydrophobicProtrusions = [];
    for idx, atom in enumerate(table) {
        if(atom.name != "CB") continue;
        if(atom not in convex_hull.vertices) continue;

        neighbourAtoms = distances.get(idx);
        neighbourAtoms = neighbourAtoms[neighbourAtoms < distance];
        if(length(neighbourAtoms) >= density) continue;

        protrusions.add(atom);
        if(isHydrophobic(atom)) hydrophobicProtrusions.add(atom)
    }
    return { protrusions, hydrophobicProtrusions };
}
```

Figure 3.3.1 - Pseudocode for finding protrusions in protein structures.
The density and distance values are given by Fuglebakk and Reuter (2018). The table argument is an ATOM record table and contains only all alpha carbon (CA) atoms and beta carbon (CB) atoms for a given protein structure (imported PDB file). The returned result contains all atoms that are protrusions and also those which are hydrophobic protrusions.

I saved the result as CSV files which you can find in the projects repository. I found all protrusions, hydrophobic protrusions and co-insertables (see chapter 2.2 for the definition of a co-insertable) for every protein structure both from the peripheral protein dataset and the reference dataset.

Not shown by the pseudocode (figure 3.3.1) is the import and filtering of the PDB files. I imported the PDB files using the Biopandas package and extracted the ATOM record table per file. I filter the ATOM record table such that there are only Cα and Cβ atoms left in the table, as the other atoms are not interesting for the project. I also filter for alternative locations, using the first alternative location with the highest occupancy.

At the end I compare my results to the results from Fuglebakk and Reuter (2018). As I show in chapter 4.1.1, I got a very close but not a perfect match to their results. Therefore I decided to investigate if there was any difference when importing and filtering PDB files. That is why you can find both a Biopandas version and a MMTK version in the project's repository. I prefer Biopandas due because it imports straight to pandas tables. Fuglebakk and Reuter (2018) used the MMTK package as mentioned before. The different packages were not the reason for the differences in the result, as reported in chapter 4.1.2.

Step 2
My next step was the visualizing of the protein convex hull and the protrusion definition. The study of proteins can be very abstract, so visualizing the problem can improve the understanding of the problem. A visualization can be used to do a visual analysis. It was important to me that I implemented a meaningful visualization that also can be used by other people later on. That is why I decided to implement a plugin for the popular molecular viewer UCSF Chimera (Pettersen *et al.*, 2004). The plugin computes and visualizes the convex hull and protrusions for a given protein of interest. The plugin gives you also the possibility to play with some variables of the protrusion definition, such as different types of convex hulls, different density and distances.

As mentioned before in the subchapter 3.2.2, I needed to adjust my algorithm from the first step, because I was not able to add the SciPy package (Millman, Jarrod Millman and Aivazis, 2011) to UCSF Chimera. I wanted to avoid a custom build of UCSF Chimera. The implementation differs from step one in the sense that I implemented my own distance computation algorithm with the Numpy package (Oliphant, 2015) and used Jinxuan Wus implementation (jinxuan, no date) of the Qhull algorithm (Barber *et al.*, 1996).

For the rendering of the convex hull the built-in engine from UCSF Chimera is used. The documentation of UCSF Chimera (*Chimera Programmer's Guide*, no date) gives a good overview about how to generate surface models and how to work with the engine.

But I needed to do some backward engineering in order to implement the graphical user interface (GUI) for my plugin. UCSF Chimera uses the tkinter module (*24.1. Tkinter — Python interface to Tcl/Tk — Python 2.7.18 documentation*, no date) for GUI elements, which is part of the python programming language. I looked into the source of several other plugins and how they work in order to figure out how I wanted to implement my GUI.

My final plugin implementation is made up of the following files. I call it the convex hull plugin, which also can show and mark protrusions.

- **ChimeraExtension.py**, which contains general information about the extension / plugin and which is initialising the plugin.

- **__init__.py**, which contains all necessary functions for the plugin to work. Here, I do all computations, update the GUI and update the viewport.

- **convexhull3d.py**, which is the Qhull implementation from Jinxuan Wu.

- **gui.py**, which contains the tkinter implementation of the GUI for the plugin and which calls different functions from the __init__.py based on the given user input.

You can find these files in the projects repository. The installation of the plugin is simple. You just need to copy the files mentioned above into a new folder, which you place inside the

UCSF Chimera’s share folder. The plugin is so available in the Tools menu, at Higher-Order Structure. I present my plugin in details in section 4.2.

Step 3
Finally, I moved on to my own study and analysis. As mentioned before I extended the work of Fuglebakk and Reuter (2018). Characterizing the environment of hydrophobic protrusions in peripheral proteins compared to the reference dataset was what I focused on in particular. Here, I began with the simplest neighbourhood definition I could come up with and added complexity as I moved on. First I looked into the neighbourhood of hydrophobic protrusions in general. Then, I looked only into the solvent neighbourhood of hydrophobic protrusions. Later on I compared the exposed neighbourhood of co-insertables to the neighbourhood of protruding, but non-protrusion hydrophobes. The resulting definitions and results are presented in chapter 4.3.

To avoid computing the same results very often, it was important to save results while I moved on. Recomputing important and time consuming features should not be necessary to do very often. To keep it simple, I decided to compute my neighbourhood feature / definition and the relative solvent accessibility (Tien *et al.*, 2013) for every residue from both datasets. This gave me about millions of lines with data which I saved in corresponding csv files. The computation of those features took about two hours for both datasets. I have now the advantage that I still had the possibility to look into these features for other amino acids, that not was interesting to look at in the first place, without recomputing. The second and more important advantage was that I just needed to read and search the data tables for my final analysis, which made the computation of minor features on the fly easily.

I worked with three feature tables for each dataset in my final analysis, the computed protrusion table, the computed relative solvent accessibility table and the computed neighbourhood table. I combined the tables as necessary. If I wanted to know *e.g.* the solvent neighbourhood of all protrusions from a protein of interest, I simply looked up all protrusions in the protrusion table for the given protein, then I looked up the neighbourhood of those protrusions in the neighbourhood table and finally looked into the relative solvent accessibility table how solvent the neighbours are, subsetting only the solvent neighbourhood of interest. Using the query function of the Pandas package (McKinney, 2010), this is done in less than a second for a protein of interest.

The computed tables are of course added to the projects repository.

Chapter 4

Results

Chapter 4.1

Finding protrusions & reproducing Fuglebakk and Reuters results

My first steps in this master project was about to reproduce the results of protrusions and hydrophobic protrusions from Fuglebakk and Reuter (2018). In order to reproduce the results I used my own implementation for all needed definitions from their paper. This gives us of course an outside point of view, which is important to remember later when comparing and evaluating the reproduced results. Reproducing the results assure correctness of my understanding about the definition of protrusions in general and gives me a good, rock solid starting point for my later study.

4.1.1 Reproducing protrusions & hydrophobic protrusions

Let us jump right on it with the following figure 4.1.1, where I compare the number of protrusions I found per peripheral protein with the number of protrusions Fuglebakk and Reuter (2018) found for the same peripheral proteins.

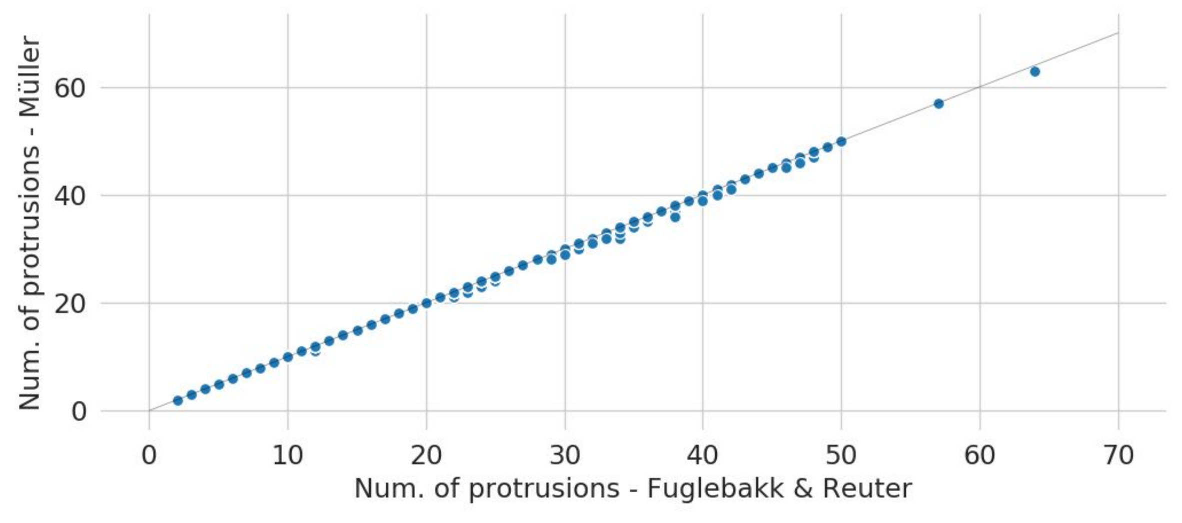


Figure 4.1.1 - The number of protrusions per protein in peripheral protein data.
This figure compares the number of protrusions for each peripheral protein that Fuglebakk and Reuter (2018) found (x-axis) with the number of protrusions I found (y-axis). The computed R^2 value is 0.9996.

The figure 4.1.1 from the previous page shows clearly that I got reproduced Fuglebakk and Reuters (2018) results for the peripheral protein data. However, we see also that not every point lies on the diagonal line. The resulting R^2 value is very close to one but not exactly one. I am underestimating Fuglebakk and Reuters (2018) results very slightly with one or two protrusions on 37 of 1012 peripheral proteins. I will explain this difference later in the subchapter 4.1.2.

The figure 4.1.2 below shows the same type of comparison for protrusions found in the reference dataset. We can see here again that not all points lie exactly on the diagonal line.

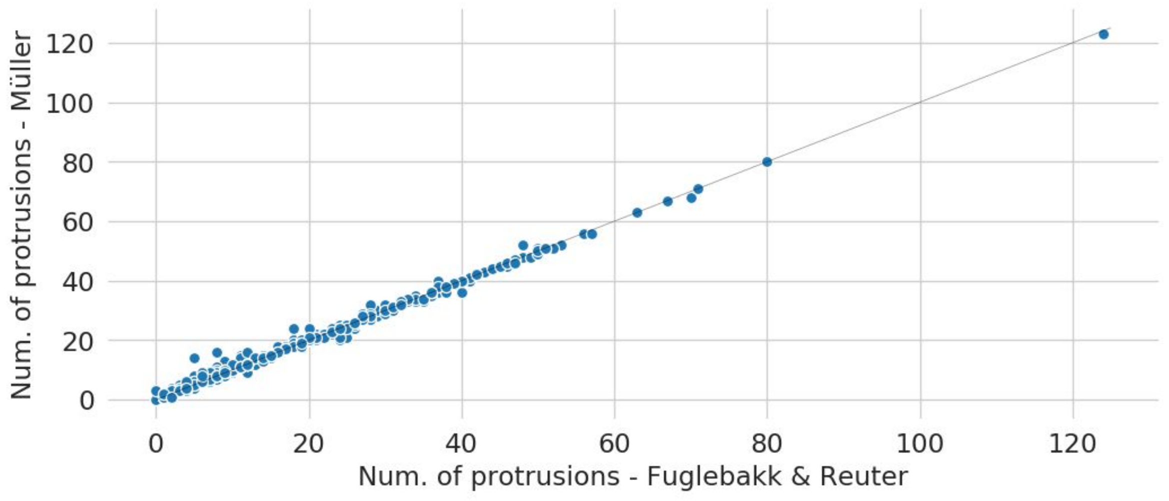


Figure 4.1.2 - Number of protrusions for each protein in the reference dataset.
This figure compares the number of protrusions for each reference protein that Fuglebakk and Reuter (2018) found (x-axis) with the number of protrusions per reference protein I found (y-axis). The computed R^2 value is 0.9946.

The R^2 value is very close to one again and I could reproduce many reference proteins with the correct number of protrusions. On reference data, however, I am both underestimating and overestimating Fulgebakk and Reuters (2018) results. I underestimated 53 of 495 reference proteins and overestimated 95 of 495 reference proteins. Both the underestimations and the overestimations are close to the diagonal line and we do not see very large outsiders. The differences are one or two protrusions in most of the cases. I conclude at this point that I reproduced the number of protrusions per reference protein from the reference dataset quite well. As mentioned before, I will explain the differences later in the subchapter 4.1.2.

Fuglebakk and Reuter (2018) provided enough information about every protrusion they found, such that I was able to evaluate if I found the same protrusions on the same proteins. When I overestimated reference proteins, then there are protrusions I found that are not confirmed by Fuglebakk & Reuter (2018). Except from those overestimated protrusions, all other protrusions I found are exactly the same protrusions on the same proteins as in Fuglebakk and Reuters (2018) results. When I underestimated a peripheral protein or reference protein, then there are one or more protrusions by Fuglebakk and Reuter (2018) that I was not able to find or confirm to be protrusions. Those are very interesting to look at in general, if there is some obvious differences and why or why not those should become protrusions or not. But more on that in the next subchapter 4.1.2.

Let us continue with a look at the distribution of hydrophobic protrusions. Hydrophobic protrusions are one of the core definitions in the paper from Fuglebakk and Reuter (2018) and important to reproduce for this study. The figure 4.1.3 shows my reproduced distribution for peripheral proteins.

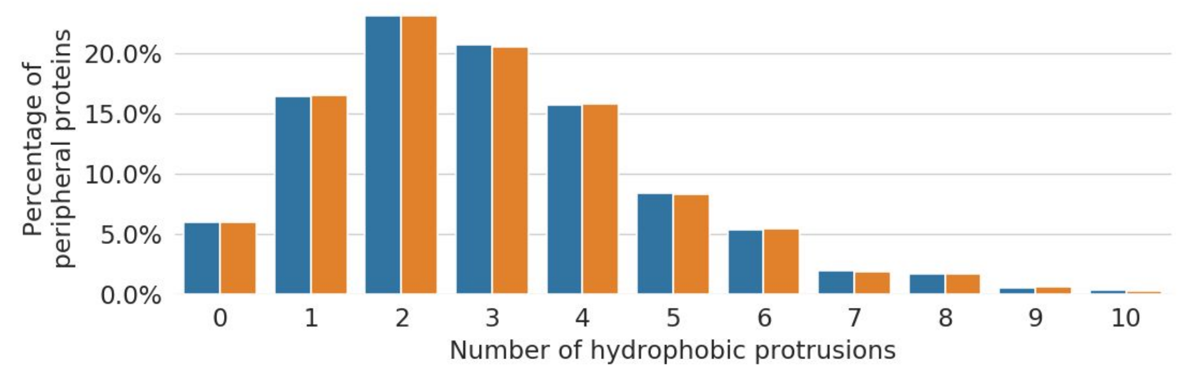


Figure 4.1.3 - Distribution of hydrophobic protrusions in peripheral protein dataset.
I show here a comparison between calculations of protrusions done with my implementation (orange bars) and the one published by Fuglebakk and Reuter (blue bars). The computed R^2 *value is 0.9986.*

I also reproduced the distribution of hydrophobic protrusions for the reference data, have a look at the figure 4.1.4 below.

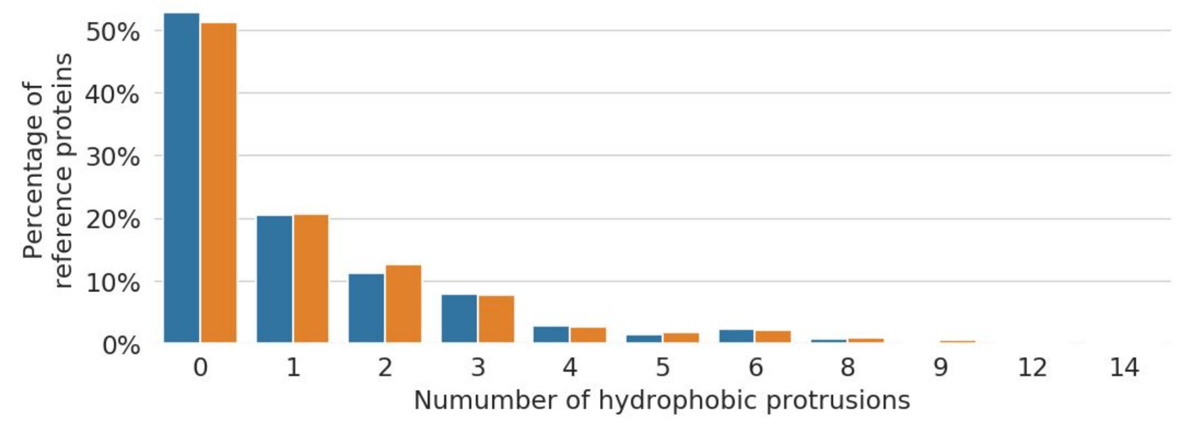


Figure 4.1.4 - ***Distribution of hydrophobic protrusions in reference dataset.***
I show here a comparison between calculations of protrusions done with my implementation (orange bars) and the one published by Fuglebakk and Reuter (blue bars). The computed R^2 *value is 0.9946.*

We can clearly observe from both the figure 4.1.3 and the figure 4.1.4 that I reproduced the distribution of hydrophobic protrusions correctly for both datasets. The R^2 values confirm that too, as the values are very close to one. What we do not see so clearly on these figures, however, are the small differences between my computations and the computations from Fuglebakk and Reuter (2018). The R^2 values are just close but not equal to one here as well. We have already seen those differences on the figures 4.1.1 and 4.1.2 and the differences still exist on this subset of protrusions.

We observe from the figure 4.1.4 that over fifty percent of all reference proteins do not contain any hydrophobic protrusion. The figure 4.1.3 shows that just a bit over five percent of all peripheral proteins do not contain any hydrophobic protrusion. The hydrophobic protrusion criteria is already separating the datasets a lot. The peripheral protein data contains 2992 hydrophobic protrusions, whereas the reference data only contains 568 hydrophobic protrusions. The difference in size is important to notice for further studies on hydrophobic protrusions with these datasets, because the reference data should not become too small later.

4.1.2 Investigating the differences

I mentioned in the introduction of chapter 4.1 that we do have an outside point of view, because I used my own implementation in order to reproduce Fuglebakk and Reuters (2018) results. It is important to understand why there are differences in order to be sure about the correctness of my results.

I am underestimating and overestimating Fuglebakk and Reuters (2018) results, as shown in the previous subchapter 4.1.1. This means that I found more protrusions (overestimating) or fewer protrusions (underestimating) than Fuglebakk and Reuter (2018) for some proteins in both datasets. My implementation differs from their implementation and that could be the reason for the differences we can observe.

The very first thing I decided to investigate was the different packages that are importing all the PDB files from both datasets. I used the Biopandas package (Raschka, 2017). Fuglebakk and Reuter (2018) used the MMTK package (Hinsen, 2000). But the imports and the filtering are identical and I still got the same differences using the old MMTK package. In order to check why this difference happens, I looked into the definition of protrusions and its criterias.

I looked first into the density criteria. Within a distance of ten angstrom (10 Å), the number of alpha carbon (Cα) atoms and beta carbon (Cβ) atoms defines the density around a protrusion (Fuglebakk and Reuter, 2018). This density has to be less than twenty-two atoms (Fuglebakk and Reuter, 2018). In order to become a protrusion, the amino acids need to fulfill the density criteria from their Cβ atom (Fuglebakk and Reuter, 2018).

Fuglebakk and Reuter (2018) provided a column in their results, called neighbours. This column shows the number of Cα and Cβ atoms they found around amino acids. I used the values from this column and compared the values to my number of neighbours I found per amino acid per protein. In my results I got the same number of neighbour atoms as Fuglebakk and Reuter (2018) for all residues. The distance criteria and the density criteria is not the reason for the differences.

In my next step I looked into the convex hull criteria. The convex hull is computed among all Cα and Cβ atom coordinates from a protein and the amino acids Cβ atom has to be a vertex of the computed convex hull in order to become a protrusion (Fuglebakk and Reuter, 2018). Fuglebakk and Reuter (2018) also provided a column for this criteria in their results, if a Cβ atom is a convex hull vertex or not. I made a corresponding column in my results and compared those to the results from Fuglebakk and Reuter (2018).

It turned out that the computed convex hull is the reason for the differences. My convex hull differs slightly on some peripheral proteins and reference proteins. The reason for underestimation are some Cβ atoms that not became part of the convex hull in my computations. The reason for overestimation is similar as some Cβ atoms became part of the convex hull in my computations but not in the computation from Fuglebakk and Reuter (2018).

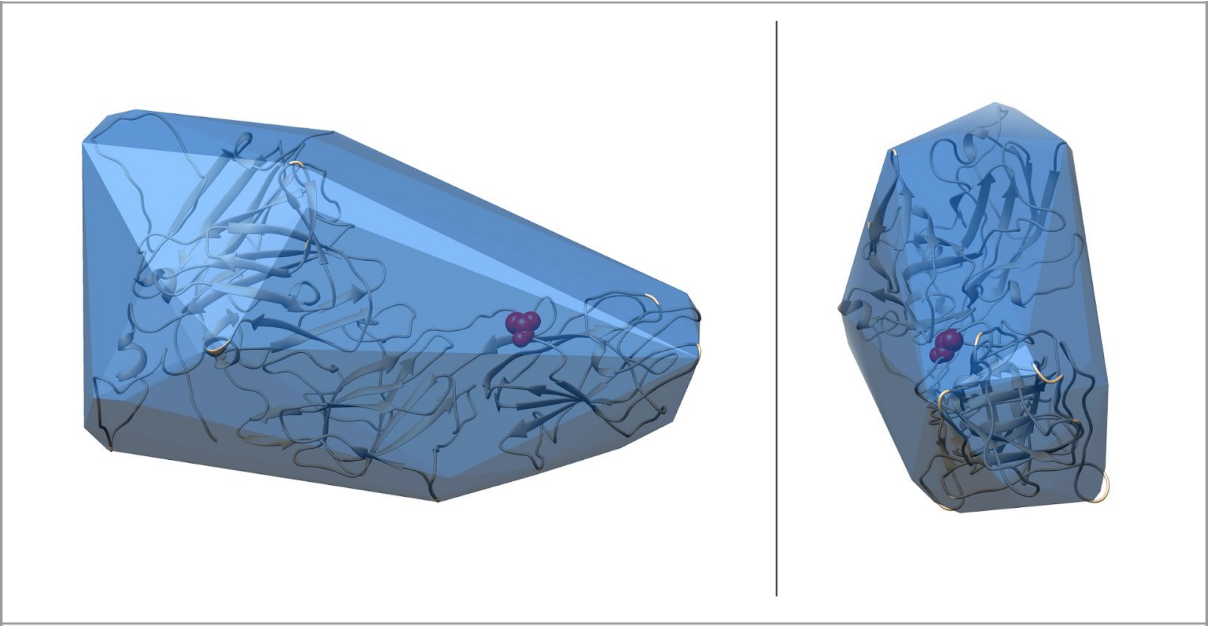


Figure 4.1.5 - Visualizing structure 3j2s with missing protrusion.
The plot shows the structure 3j2s, which I underestimated in my computations. The structure is shown from two different perspectives. The computed convex hull is blue and the atoms of the missing protrusion residue are marked as red balls. The missing protrusion is ALA-2184 on chain B.

I visualized an example with UCSF Chimera (Pettersen *et al.*, 2004) and my own convex hull plugin I created for UCSF Chimera. You can read more about protrusion visualization and my plugin in the next chapter 4.2. The figure 4.1.5 above shows the peripheral protein 3j2s. I underestimated the structure by one protrusion. My computed convex hull is shown in blue and the missing protrusion residue ALA-2184 is marked in red. We can clearly observe that the missing protrusion is quite a bit inside of the computed convex hull. I can also imagine that the missing protrusion can be part of the convex hull. My convex hull looks more general and not close enough to the structure in order to fit Fuglebakk and Reuters (2018) computations for this peripheral protein.

After visualizing other proteins with differences from both the peripheral protein dataset and the reference dataset, the structure 3j2s seems to be more or less representative enough. We can conclude that the reason for the differences is caused by different convex hull computations and we got an impression how that can look like.

One possible reason could be the convex hull algorithm itself. Fuglebakk and Reuter (2018) used Scipy's (Millman, Jarrod Millman and Aivazis, 2011) implementation of Qhull (Barber *et al.*, 1996) and so did I. But I used a newer version of the software. Minor changes to the software among versions may explain different results. However, my convex hull plugin for UCSF Chimera uses Jinxuan Wus implementation (jinxuan, no date) of the Qhull algorithm (Barber *et al.*, 1996). I noticed no differences between my results using Scipy's and Jinxuan Wus implementation of the Qhull algorithm.

There is also the possibility to use the Qhull algorithm with several, additional arguments (Barber *et al.*, 1996). I tried different combinations of arguments for a tighter convex hull, but I was not able to reproduce Fuglebakk and Reuters (2018) results at a hundred percent match. I got the best match when not using any additional arguments at all, as shown in subchapter 4.1.1. That of course does not exclude the possibility that Fuglebakk and Reuter (2018) used some additional, unknown arguments.

One last possibility that I want to come up with is the protein data. There is a possibility that the proteins in the database changed such that atoms got different coordinates. This could result in different convex hulls too. Fuglebakk and Reuter (2018) provided for all protrusions the coordinates of the corresponding Cα atom in their data. I checked all Cα atom coordinates for each protein against my data and the coordinates of any Cα atom are the same. This excludes not any other atom coordinate changes, but if there are changes, the changes will not be found without validating against Fuglebakk and Reuters (2018) original files they used.

At the end of this chapter, I summarize that I got reproduced Fuglebakk and Reuters (2018) results quite good from my outside point of view. As shown in the subchapter 4.1.1, I matched the number of protrusions and the distributions of hydrophobic protrusions quite well for both datasets. The consistency is good in general. I consider this reproduction as good enough to assure correctness of my understanding about the definition of protrusions and that I am able to apply the definition the correct way.

The differences between my results and Fuglebakk and Reuters (2018) results, as shown in chapter 4.1.1, can be explained with slightly different convex hull computations. The reason for that stays unknown by now but newer software versions and changes of proteins in the database could be possible explanations.

Chapter 4.2

Visualizing protein protrusions with UCSF Chimera

As mentioned many times before, I implemented a convex hull plugin / extension for the popular molecular viewer UCSF Chimera (Pettersen *et al.*, 2004). The convex hull plugin is not only computing and showing the convex hull in real time, but also gives you the possibility to show protrusions and to play with some of the variables / arguments used in the computations. It is now possible to study peripheral proteins and the definition of protrusions in a more interactive, visual way.

The installation process is already mentioned in section 3.3.1, step 2. The plugin contains the following four files, which you place inside a folder within UCSF Chimeras share folder. You can find these files in the projects repository.

- **ChimeraExtension.py**, which contains general information about the extension / plugin and which is initialising the plugin.

- **__init__.py**, which contains all necessary functions for the plugin to work. Here, I do all computations, update the GUI and update the viewport.

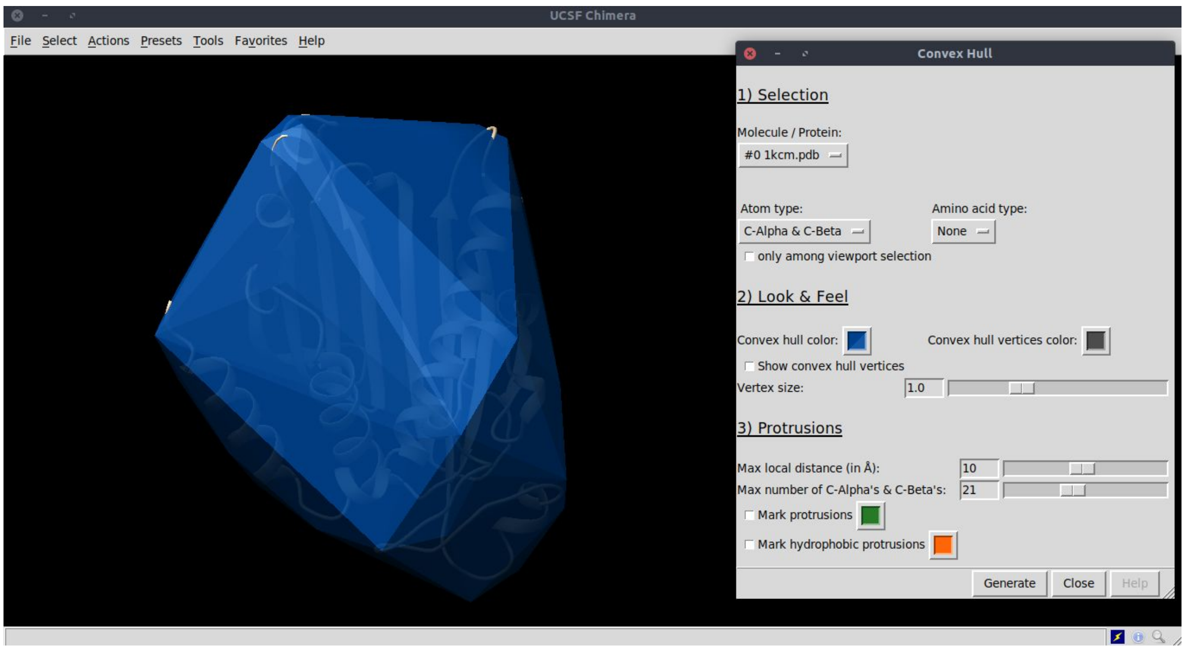
- **convexhull3d.py**, which is the Qhull implementation from Jinxuan Wu.

- **gui.py**, which contains the tkinter implementation of the GUI for the plugin and which calls different functions from the __init__.py based on the given user input.

Then, starting USCF Chimera, the plugin will be recognized and accessible from the Tools menu, at the submenu Higher-Order Structure.

4.2.1 Overview of the convex hull plugin

Starting the convex hull plugin will open up an additional window with the plugins graphical user interface (GUI). If you already loaded a protein into UCSF Chimera, then it will be recognized by the plugin. A convex hull will be computed and shown immediately. If there are multiple protein structures present, the plugin takes the first structure by default. If you want to change the current protein structure to work with, simply select the molecule / protein of interest from the first drop down menu of the selection part.



1) Selection

Molecule / Protein: #0 1kcm.pdb

Atom type: C-Alpha & C-Beta

Amino acid type: None

☐ only among viewport selection

2) Look & Feel

Convex hull color: 🟦

Convex hull vertices color: ⬛

☐ Show convex hull vertices

Vertex size: 1.0

3) Protrusions

Max local distance (in Å): 10

Max number of C-Alpha's & C-Beta's: 21

☐ Mark protrusions 🟩

☐ Mark hydrophobic protrusions 🟧

Figure 4.2.1 - Screenshot of UCSF Chimera and the convex hull plugin
The convex hull is applied to the peripheral protein 1kcm with the plugins default arguments.

The GUI is divided into three sections, the selection, the look & feel and the protrusion section. In the first section it is possible to modify the atom selection, which the convex hull computations are based on. Fuglebakk and Reuter (2018) computed the convex hull among all alpha carbon atoms (Cα) and beta carbon atoms (Cβ) of the protein and this is of course possible with my plugin too. In addition it is possible to compute a convex hull based on just Cα atoms or Cβ atoms.

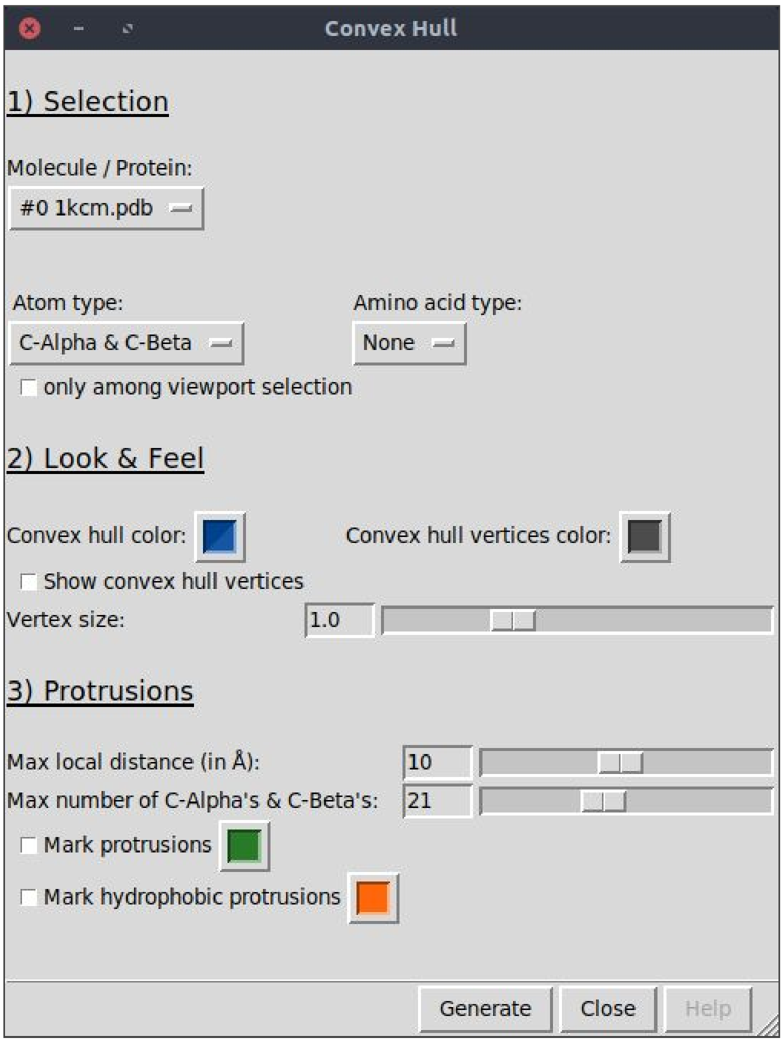
One can also restrict the selection to a group of amino acid residues of interest. By doing so it is possible to compute a convex hull only among all hydrophobic, aromatic, positively charged or negatively charged amino acids of the structure. If the amino acid type selection is none, then all amino acids regardless of their type will be considered for the convex hull computation.

Finally, you may want to restrict the selection only to your own viewport selection in order to compute a special convex hull. In this case, simply do your viewport selection and check the “only among viewport selection” checkbox. If necessary, one can also combine certain types of selections, *e.g.* all aromatic amino acids among the viewport selection. Any changes becomes visible in real time for the user.

The look & feel part gives the user the possibility to change the color of the convex hull with a color picker. It also gives the user the possibility to show the atoms that became the convex hull vertices. The convex hull vertex color can be changed as well if shown. In order come by different structures and their size, the user can use a slider to adjust the vertex size if needed. This will affect all visible atoms, which gives the user a uniform look of the protein.

The last part of the GUI is about the protrusion definition. The user has the ability to colorized all convex hull vertices that are protrusions or hydrophobic protrusions. The user can also change the color of protrusion atoms of course. The GUI provides two sliders, which the user may use to play with the protrusion criterias. The user can adjust both the distance and the max number of neighbour Cα and Cβ atoms. Every change the user takes here will be visible immediately as well.

I provide a detailed screenshot of the convex hull window below, figure 4.2.2, which shows the defaults right after I opened the window. I also provide another screenshot, figure 4.2.3, which shows the plugin in a use case scenario. Here, some of the options are modified and all computed protrusions are visible and colorized for the structure 1kcm.


Convex Hull

1) Selection

Molecule / Protein: #0 1kcm.pdb

Atom type: C-Alpha & C-Beta

Amino acid type: None

☐ only among viewport selection

2) Look & Feel

Convex hull color:

Convex hull vertices color:

☐ Show convex hull vertices

Vertex size: 1.0

3) Protrusions

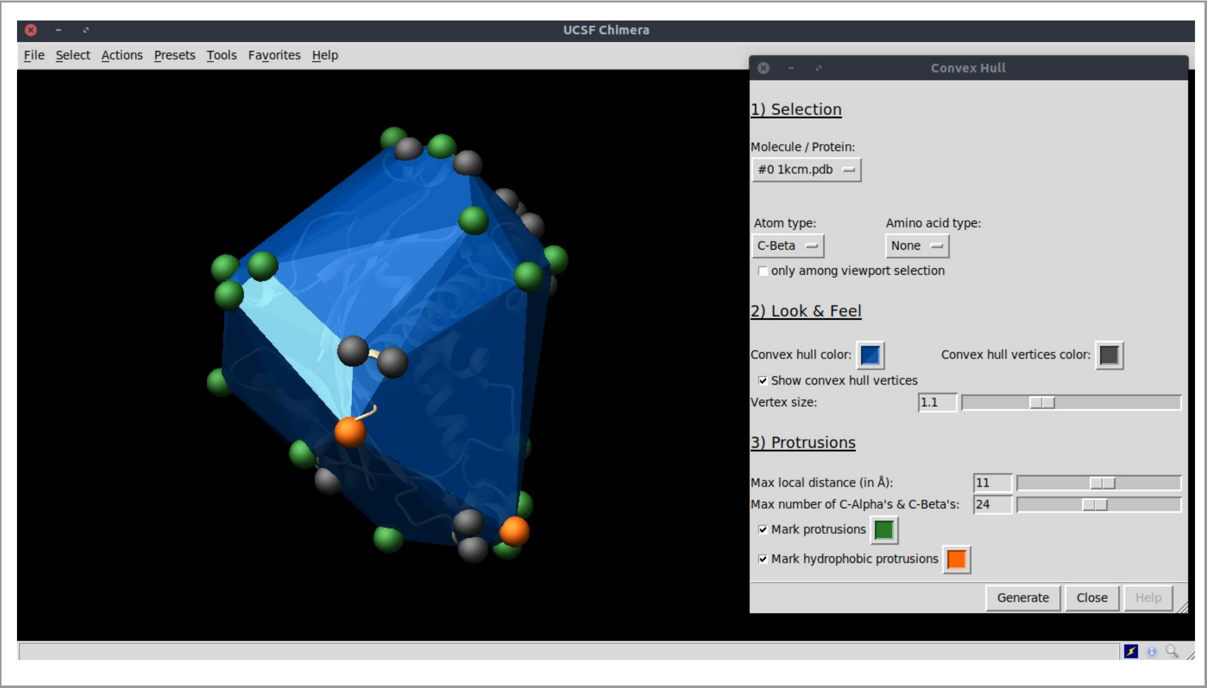
Max local distance (in Å): 10

Max number of C-Alpha's & C-Beta's: 21

☐ Mark protrusions

☐ Mark hydrophobic protrusions

Figure 4.2.2 - Screenshot of the convex hull plugin window only
The figure shows only the convex hull plugin window for a detailed look at the options. The default values are applied to the peripheral protein 1kcm (see figure 4.2.1).



1) Selection

Molecule / Protein: #0 1kcm.pdb

Atom type: C-Beta

Amino acid type: None

☐ only among viewport selection

2) Look & Feel

Convex hull color: 🟦

Convex hull vertices color: ⬛

☑ Show convex hull vertices

Vertex size: 1.1

3) Protrusions

Max local distance (in Å): 11

Max number of C-Alpha's & C-Beta's: 24

☑ Mark protrusions 🟩

☑ Mark hydrophobic protrusions 🟧

Figure 4.2.3 - Screenshot of UCSF Chimera and the convex hull plugin showing protrusions
The convex hull is applied to the peripheral protein 1kcm. Some arguments of the convex hull and the protrusion definition has been customized. The convex hull vertices are visible with the colors of choice. In this case convex vertices are grey, protrusions are green and hydrophobic protrusions are orange.

Finally, the user simply clicks on the “Generate” button in order to keep and finalize the current convex hull and protrusion visualization. The plugin window can so be closed. If the user closes the plugin window without clicking the generate button, the temporary convex hull will be removed again.

4.2.2 About performance

I mentioned in the previous section that changes in the graphical user interface (GUI) made by the user become visible immediately in real time. The performance is fast enough and not an issue at all if you run my convex hull plugin on most low spec laptops for most, small proteins. In detail, it is actually the distance computation and then the rendering of the convex hull that takes most of the computation time.

However, some changes can take two seconds, three seconds or even longer to compute if the protein is very large and with a lot of atoms. In such cases I remove all GUI elements and show a loading screen instead. As soon as the computation is done, the GUI elements becomes visible again. I implemented this loading screen to prevent the user from doing more user inputs while the computation of the last input is running.

On Apple computers, the GUI just freezes for user inputs. Just freezing the GUI is very common from other USCF Chimera plugins unfortunately. Users of USCF Chimera are familiar with such type of experience. I did not find out why my loading screen is not shown on Apple computers, as I not had access to a mac and was not able to investigate the problem. But I noticed that my tutor has not experienced any loading screen running the plugin on a Mac computer.

4.2.3 License

As I implemented a plugin for a existing software, it is also important to reflect both about the license of the software and the license I can provide for my plugin. UCSF Chimera declares in its license that it can freely be installed and used for academic, non-profit, or government-sponsored research purposes (*UCSF Chimera Non-Commercial Software License Agreement*, no date). However, a commercial use of the software is not permitted and needs separate licenses.

Further on UCSF Chimera writes in their license:
"*Licensee agrees that it will use the Software, and any modifications, improvements, or derivatives to the Software that the Licensee may create (collectively, "Improvements") solely for internal, non-commercial purposes and shall not distribute or transfer the Software or Improvements to any person or third parties (...)*
The term "non-commercial," as used in this License, means academic or other scholarly research which (a) is not undertaken for profit, or (b) is not intended to produce works, services, or data for commercial use, or (c) is neither conducted, nor funded, by a person or an entity engaged in the commercial use, application or exploitation of works similar to the Software." (*UCSF Chimera Non-Commercial Software License Agreement*, no date)

As I understood the license of UCSF Chimera, I am able to distribute my work as long as it is non-profit and / or part of academic work. I decided that my plugin should be open source and everyone is free to copy, modify or share my plugin work for non-commercial use within the UCSF Chimera non-commercial software license.

Chapter 4.3

The neighbourhood of exposed hydrophobes and hydrophobic protrusions

What types of amino acids can we find around hydrophobic protrusions? I really like the simplicity of this question. We may get useful information about potential membrane binding sites of peripheral proteins. Why?

We know that biology can be very complex and so is peripheral protein binding to the membrane. Fuglebakk and Reuter (2018) showed that a model of protruding, co-insertable hydrophobes can be used to distinguish surfaces of peripheral membrane proteins from the surfaces of transmembrane proteins in a reference dataset. They also showed that hydrophobic protrusions can be used to identify potential membrane-binding residues of peripheral proteins. In order to identify membrane-binding residues they proposed the concept of the *likely inserted hydrophobe*, which is defined as the protruding hydrophobe with the lowest local protein density and the highest number of other protruding hydrophobes in its close neighbourhood. Then they showed that the likely inserted hydrophobe coincides with the binding sites of known peripheral proteins (Fuglebakk and Reuter, 2018).

But known binding domains from peripheral proteins are made up by more than just one binding site (Luckey, 2014). Luckey (2014) described typical membrane-binding domains such as C1, C2, FYVE and PH. She described that each domain type has a specific binding site that is usually for a lipid ligand, but she also described that a specific binding site is often supported by additional, less specific binding sites. Known binding sites are mainly made of hydrophobic amino acids and positive charged amino acids (Luckey, 2014). Fuglebakk and Reuter (2018), however, considered only the other hydrophobic protrusions (co-insertables) in their model. Therefore, a look at what kind of residues we can find around hydrophobic protrusions is now necessary. It will inform about the general protein surface composition around hydrophobic protrusions, that may bind more likely to membranes.

4.3.1 The neighbourhood

In order to rigorously analyse the protein surface around hydrophobic protrusions, I find it important to clearly define the neighbourhood of hydrophobic protrusions. The choice of the distance criteria is critical. We need to archive a neighbourhood that is representative enough. At this point it is useful to remember the definition of hydrophobic protrusions. Among other properties, Fuglebakk and Reuter (2018) used a distance criteria and a density criteria in their definition. They used a distance of ten angstrom (10 Å) and within this distance the number of alpha carbon (Cα) atoms and beta carbon (Cβ) atoms defines the density around a hydrophobic protrusion. We can adopt and extend this concept to define the neighbourhood of hydrophobic protrusions as the amino acids to which those Cα and Cβ atoms belong to. This definition has the advantage of being consistent with the previous study from Fuglebakk and Reuter (2018).

Definition 4.3.1 - The neighbourhood of a hydrophobic protrusion
The neighbourhood of a hydrophobic protrusion is defined as the amino acids whose Cα and Cβ atoms are found within a distance of 10 Å of the protrusion, but the hydrophobic protrusion residue itself is excluded.

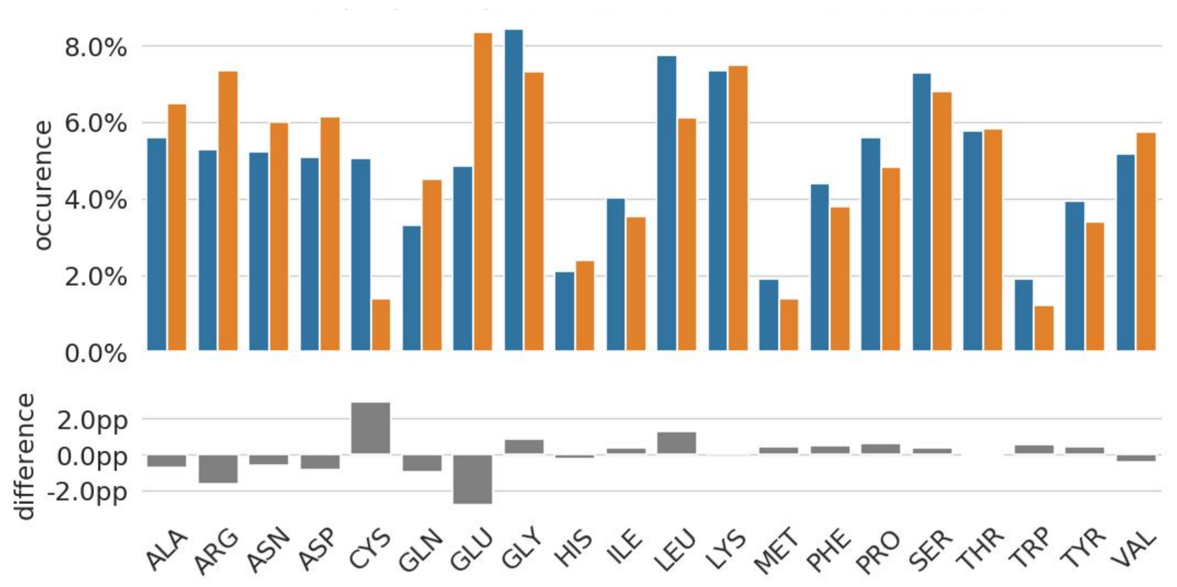


Figure 4.3.1 - The neighbourhood of hydrophobic protrusions.
I first computed all hydrophobic protrusions. Then I computed the corresponding neighbourhood for each of them. I merged all neighbourhoods together in one large data table and finally normalized to the relative frequencies of unique amino acid residues. The first (upper) part of this figure shows the relative frequencies for both the peripheral protein dataset (blue bars) and the reference dataset (orange bars). The second (lower) part of this figure shows the difference of the relative frequencies between the two datasets in percent points (pp). Here, a positive value means that there is a higher relative frequency for a particular amino acid in the peripheral protein dataset.

Using my definition of the neighbourhood, the figure 4.3.1 on the previous page shows how common the twenty types of amino acids are in neighbourhoods of hydrophobic protrusions. First of all it is interesting to see that all twenty residues are indeed present on the figure. No single amino acid type is predominant.

We can *e.g.* see that histidine (HIS), methionine (MET) and tryptophan (TRP) are not very common neighbours of hydrophobic protrusions in general. Glycine (GLY), lysine (LYS) and serine (SER) seem to be more common neighbours. The figure shows clearly some major differences between datasets too. Cysteine (CYS) and leucine (LEU) are more present in neighbourhoods of hydrophobic protrusions from the peripheral protein dataset, whereas in the reference dataset, glutamic acid (GLU) and arginine (ARG) are more common neighbours than in the peripheral protein dataset. We also observe that there is almost no difference between datasets when we look at lysine (LYS) and threonine (THR).

We now have a general impression about amino acid residues in neighbourhoods of hydrophobic protrusions. Let us move on by looking at neighbourhoods from a more functional point of view. Instead of looking at residue type frequencies, it would be more interesting to look at biochemical and biophysical properties. The number of residues in a neighbourhood having a certain property, divided by the total number of residues in the neighbourhood, should inform about the properties of the neighbourhood of hydrophobic protrusions. The neighbourhood's hydrophobicity and charge are in particularly highly interesting to look at.

Definition 4.3.2 - The neighbourhoods hydrophobicity and charge
The hydrophobicity of a neighbourhood is computed as the number of hydrophobic amino acids in the neighbourhood divided by the total number of residues in that neighbourhood. Hydrophobic residues are cysteine (CYS), isoleucine (ILE), leucine (LEU), methionine (MET), phenylalanine (PHE), tryptophan (TRP) and tyrosine (TYR), according to the Wimley and White (1996) hydrophobicity scale at membrane interfaces.
The positive charge of a neighbourhood is computed as the number of positively charged amino acids in the neighbourhood divided by the total number of residues in that neighbourhood. Positive charged residues are arginine (ARG), histidine (HIS) and lysine (LYS).
The negative charge of a neighbourhood is computed as the number of negatively charged amino acids in the neighbourhood divided by the total number of residues in that neighbourhood. Negative charged residues are aspartic acid (ASP) and glutamic acid (GLU).

The hydrophobicity of the neighbourhood seems reasonable and almost obvious because of the membrane biology, the peripheral proteins biology and the definition of a hydrophobic protrusion. An idea is that a hydrophobic residue may prefer other hydrophobic residues around. A single hydrophobic residue may not be a stable binding site, having Fuglebakk and Reuters (2018) definition of co-insertables in mind. Anyway, only looking for other hydrophobic protrusions (co-insertables) may not be enough. Looking for more hydrophobicity in the neighbourhood as well is absolutely the way to go.

The electrostatic properties might play a important role when peripheral proteins bind to the membrane (Luckey, 2014). We know that some amino acids are charged. We know that binding sites of peripheral proteins can be made of positively charged amino acids (Johnson and Cornell, 1999). We also know that lipids can be charged too (Luckey, 2014). That knowledge gives me enough reasons to look into both the positive charge and the negative charge of a protrusion’s neighbourhood.

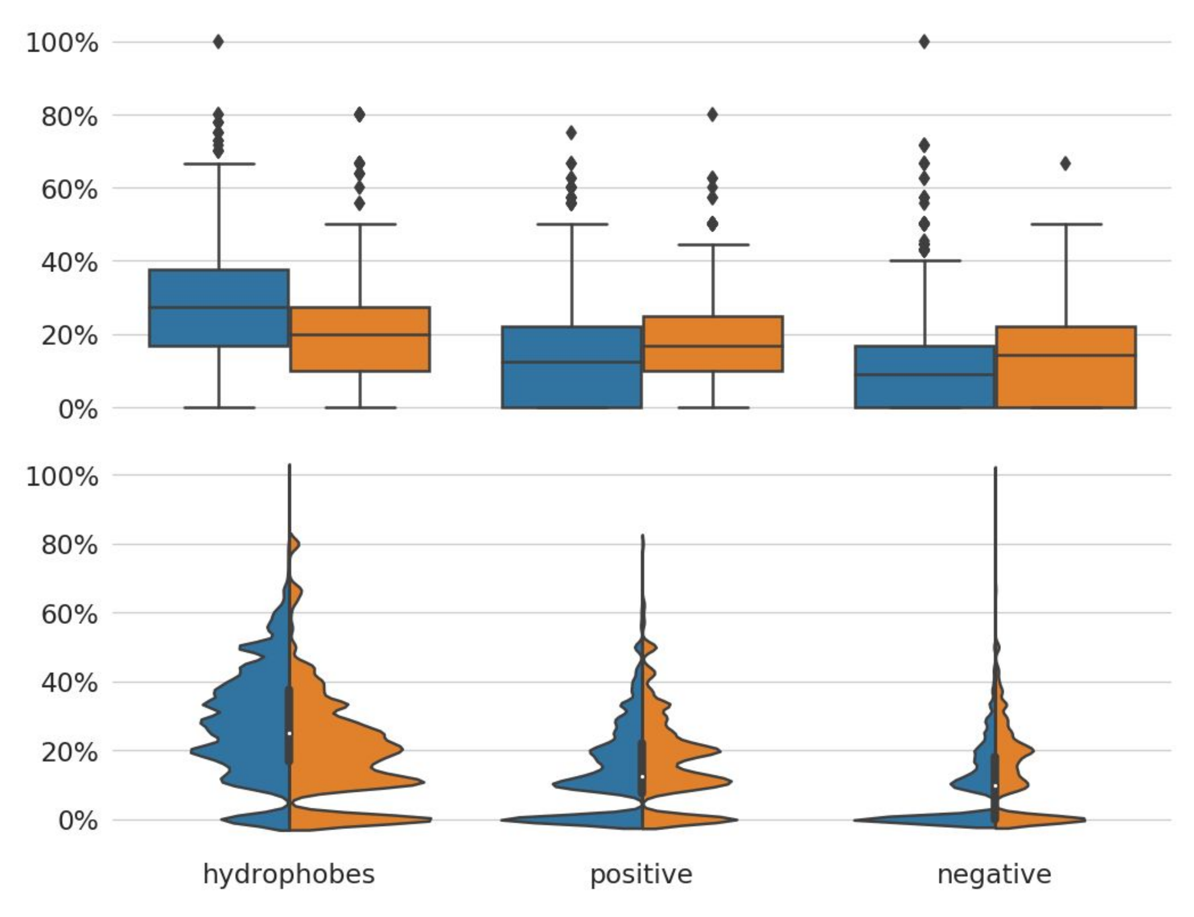


Figure 4.3.2 - The neighbourhood's hydrophobicity and charge.
The first (upper) part shows the computed hydrophobicity and charge as a boxplot. The second (lower) part of this figure shows the corresponding violinplot to get an idea about the distributions. The peripheral protein dataset is represented in blue and the reference dataset is represented in orange.

Using my definition 4.3.2, the figure 4.3.2 shows the computed hydrophobicity and charge for all computed neighbourhoods. The figure contains both a boxplot and a corresponding violinplot for hydrophobicity and charge of the hydrophobic protrusion neighbourhood . The boxplot gives us useful information about how the properties are spread out in the neighbourhood of hydrophobic protrusions. The boxplot can still be misleading as the boxplot is not affected by the distribution of the properties. That is the reason why I also show a violinplot that shows us the corresponding density functions for all properties of interest. We can now see the spread, the confidence intervals, outsiders, medians and the density function (frequency) for a property of interest.

We can clearly see that most neighbourhoods of hydrophobic protrusions do have some significant hydrophobicity. The hydrophobicity of neighbourhoods from the datasets lies mainly between ten and forty percent. We can also see that neighbourhoods from the peripheral protein data are roughly ten percent points more hydrophobic and slightly more spread than neighbourhoods from the reference data. The distributions on the violinplot confirms that there are significantly fewer hydrophobic neighbours in neighbourhoods from the reference data than in neighbourhoods from the peripheral protein data. It is also worth to notice that the zeroness for the hydrophobicity of neighbourhoods from the reference data is greater than the zeroness of neighbourhoods from the peripheral protein data. Many hydrophobic protrusions from the reference data have no hydrophobic neighbour at all.

The figure 4.3.2 shows also that neighbourhoods can have a charge. Neighbourhoods with their origin from the reference data are slightly more charged than neighbourhoods from the peripheral protein data. The distributions on the violinplot for both the positive charge and the negative charge shows large peaks at zero for neighbourhoods from both datasets. A large number of neighbourhoods are not positive or negative, or neither positive nor negative. The positive charge of a neighbourhood seems to be more common than the negative charge, especially for neighbourhoods from the reference data.

That being said it is relevant to combine hydrophobicity with charge in order to understand how these properties interact with each other. I took every neighbourhood with a non-zero hydrophobicity and found how many positive and negative amino acids they contain. By doing so we can evaluate which neighbourhoods are charged and which are neutral. Neighborhoods with more positives than negatives gives positively charged neighbourhoods and vice-versa. Neighborhoods with equal numbers of positives and negatives are neutral.

On the next page, the following table 4.3.1 shows the resulting neighbourhood frequencies, combining hydrophobicity and charge. We can observe that neighbourhoods with non-zero hydrophobicity are more frequently positively charged than negatively charged.

Class / Combination / Condition	Periph. protein data neighbourhoods	Reference data neighbourhoods
Hydrophobicity ∩ (Positives > Negatives)	43.05%	35.21%
Hydrophobicity ∩ (Positives < Negatives)	23.33%	27.29%
Hydrophobicity ∩ (Positives == Negatives > 0)	15.88%	15.85%
Hydrophobicity ∩ (Positives == Negatives == 0)	9.96%	5.28%
Other (with no hydrophobicity)	7.78%	16.37%

Table 4.3.1 - Charge properties of neighbourhoods with non-zero hydrophobicity
The table shows how frequent neighbourhoods with a non-zero hydrophobicity are charged or neutral. The table also contains the percentage of other neighbourhoods with no hydrophobicity.

Over sixty percent of all hydrophobic neighbourhoods are charged in both datasets. More than forty percent of all neighbourhoods from peripheral protein data with a non-zero hydrophobicity do have more positive charged amino acids than negative ones. The number of negatively charged neighbourhoods in peripheral protein data is slightly less than in the reference data. There is no large difference between data sets in the frequency of neutral neighborhoods. We also see that the number of neutral neighbourhoods is higher in the peripheral protein data than in the reference data.

It is interesting to see from the previous figure 4.3.2 that neighbourhoods of hydrophobic protrusions from the reference data are more positively charged than neighbourhoods from the peripheral protein data. However, when we are combining hydrophobicity and charge, we see clearly that neighbourhoods from the peripheral protein data are about seven percent points more positively charged among hydrophobic neighbourhoods. The slightly higher positively charge of neighbourhoods from the reference data comes from non-hydrophobic neighbourhoods.

To summarise this subchapter we can remember that all twenty residues can be present in neighbourhoods of hydrophobic protrusions. Histidine (HIS), methionine (MET) and tryptophan (TRP) are not very common neighbours of hydrophobic protrusions in general but glycine (GLY), lysine (LYS) and serine (SER) seem to be more common. Cysteine (CYS) and leucine (LEU) are more present in neighbourhoods of hydrophobic protrusions in peripheral protein data than in the reference dataset. In the reference dataset, glutamic acid (GLU) and arginine (ARG) are more common neighbours than in the peripheral proteins.

In peripheral proteins we observe that 92.22% of all hydrophobic protrusions have one or more hydrophobic neighbours. Their neighbourhoods are also positively charged (more or less) in over forty percent of the cases. In the reference dataset we observe that hydrophobic protrusions have less hydrophobes in their neighbourhoods and their neighbourhoods tend to be more charged in general.

The core of biological membranes is hydrophobic while their surface is negatively charged as lipids can have neutral or negatively charged heads (Luckey, 2014). Taking this as a foundation it absolutely makes sense that neighbourhoods of hydrophobic protrusions on peripheral proteins try to fulfill compatible properties in being hydrophobe (to be able to enter the membrane) and positive charged (to be attracted from negative charged lipids in the membrane). But we see the same trends of properties on hydrophobic protrusions in the reference data too. However, hydrophobicity and the positive charge in a combination seems to be more distinct in neighbourhoods of hydrophobic protrusions from protein peripheral data than in neighbourhoods from the reference data.

4.3.2 The exposed neighbourhood

We have until now defined the neighbourhood of a hydrophobic protrusion as every amino acid within a distance of 10 Å from the protrusion, as described in chapter 4.3.1. This definition may include residues that are not exposed on the surface of the protein. Such residues are unlikely to play a role in protein membrane binding, because their position is too deep in the protein structure and not accessible by the membrane lipids. To refine our first analysis, we now turn to analyse only the surface-exposed amino acids in neighborhoods of hydrophobic protrusions. Computing the relative solvent accessibility (RSA) for a given residue to define its level of surface-exposure is a common approach in protein analysis (Tien *et al.*, 2013).

To compute the RSA, the amino acids solvent accessibility surface area (SASA) needs to be normalized by a reference value (Tien *et al.*, 2013). Without going into too much details at this point, Tien et al (2013) published a table showing theoretical, maximum possible solvent accessible surface area (maxSASA) for every of the twenty amino acids. The theoretical maxSASA values in this table have been obtained from computationally built Gly-X-Gly tripeptides, where X is the amino acid residue of interest (Tien *et al.*, 2013). They also compared their maxSASA values to the earlier methods from Miller et al. (1987) and Rose et al. (1985). Summarized, the RSA is computed as the residues SASA divided by the maxSASA of that type of residue (Tien *et al.*, 2013). I computed the residues SASA with Mitternachts (2016) FreeSASA software.

It is important that we define a level of surface exposure and a meaningful threshold for the RSA value at this point. A threshold of twenty percent is used in Chen and Zhous (2005) publication about the prediction of solvent accessibility. Based on that, I decided that a residue is exposed if the RSA is equal to or greater than twenty percent. Twenty percent is a good threshold, because we want to include residues that are just little exposed. In protein membrane binding, the binding site of peripheral proteins enters the membrane (Luckey, 2014), which indicates that also little exposed residues could be included in potential binding sites.

Let us extend my neighbourhood definition from the previous chapter 4.3.1 with the RSA criteria and let us make the same analysis as in chapter 4.3.1 for the exposed neighbourhoods of hydrophobic protrusions.

Definition 4.3.3 - The definition of a exposed residue and the exposed neighbourhood

An amino acid residue is exposed if the computed relative solvent accessibility (RSA) for that residue is equal to or larger than 20%.

The exposed neighbourhood of a hydrophobic protrusion is defined as all exposed amino acids whose Cα and Cβ atoms are found within a distance of 10 Å of the protrusion. The hydrophobic protrusion residue itself is excluded.

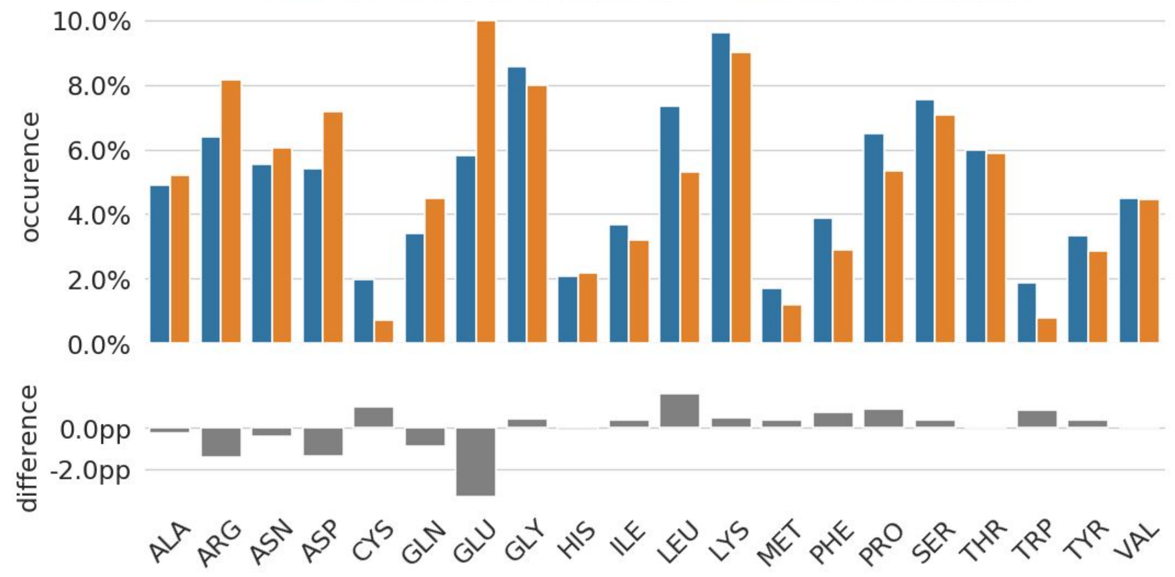


Figure 4.3.3 - The exposed neighbourhood of hydrophobic protrusions.
I first computed all hydrophobic protrusions. Then I computed the corresponding neighbourhood for each of them. I computed the RSA for each amino acid residue in all neighbourhoods, then I filtered out residues with RSA below twenty percent. I merged all exposed neighbourhoods together in one large data table and finally normalized to the relative frequencies of unique amino acid residues. The first (upper) part of this figure shows the relative frequencies for both the peripheral protein dataset (blue bars) and the reference dataset (orange bars). The second (lower) part of this figure shows the difference of the relative frequencies between the two datasets in percent points (pp). Here, a positive value means that there is a higher relative frequency for a particular amino acid in the peripheral protein dataset.

We observe from the figure 4.3.3 that all twenty residues are present in the exposed neighbourhood of hydrophobic protrusions as well. Compared to the figure 4.3.1 from the previous subchapter 4.3.1, we can see some differences. In order to highlight all changes, I made the following figure 4.3.4.

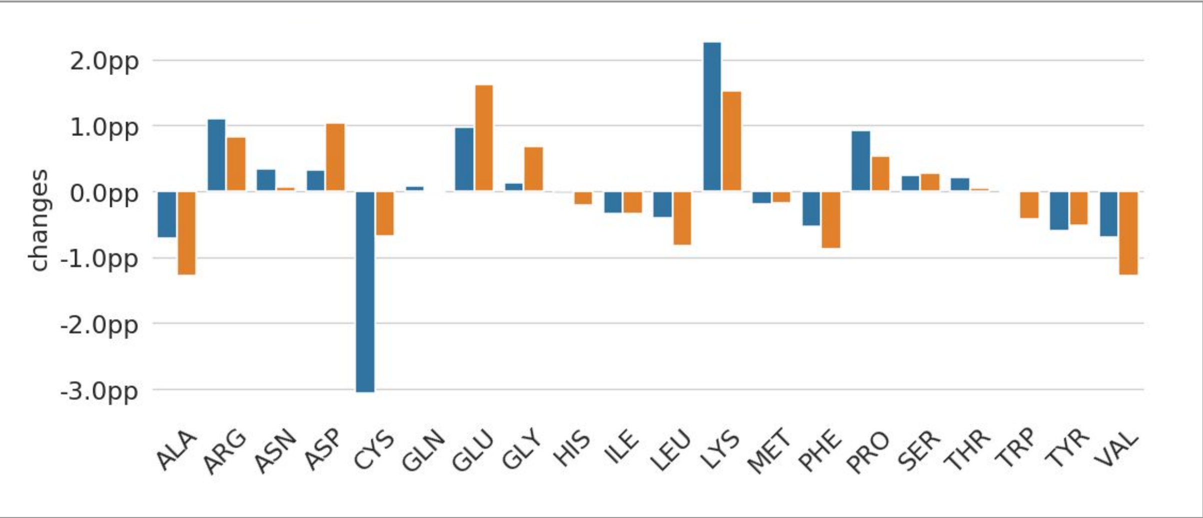


Figure 4.3.4 - Changes from the whole neighbourhood to the exposed neighbourhood
This figure shows the changes of the barplot from figure 4.3.1 (The neighbourhood of hydrophobic protrusions) to the figure 4.3.3 (The exposed neighbourhood of hydrophobic protrusions) in percent points (pp). Changes for the peripheral protein dataset are represented as blue bars and changes for the reference dataset are represented as orange bars.

Both comparing the figure 4.3.3 with the figure 4.3.1 and the look at the figure 4.3.4 shows that *e.g.* arginine (ARG), glutamic acid (GLU) and lysine (LYS) are more common neighbours in the exposed neighbourhood than in the overall neighbourhood. The presence of lysine (LYS) in the exposed neighbourhood increased with over two percent points for the peripheral protein data (figure 4.3.4) and reaches over eight percent for both datasets (figure 4.3.3). We can also observe that *e.g.* alanine (ALA), cysteine (CYS) and valine (VAL) are less common neighbours in the exposed neighbourhood than in the overall neighbourhood. The presence of cysteine (CYS) decreased with three percent points in the exposed neighbourhood for the peripheral protein data (figure 4.3.4) and its presence is below two percent for both datasets now (figure 4.3.3).

We remember from the figure 4.3.1, that cysteine (CYS) and glutamic acid (GLU) are those amino acids with the largest differences between datasets in whole neighbourhoods. The large difference of cysteine (CYS) between datasets is not that strong anymore in the exposed neighbourhood (figure 4.3.3). Leucine (LEU) takes over this part as a more common amino acid and with the second highest difference between datasets, being more present in the peripheral protein data. The difference of glutamic acid (GLU) increased on the other hand. Glutamic acid (GLU) seems to be a very common neighbour in the exposed neighbourhood, especially for exposed neighbourhoods from the reference data where glutamic acid (GLU) makes up roughly ten percent of all neighbours.

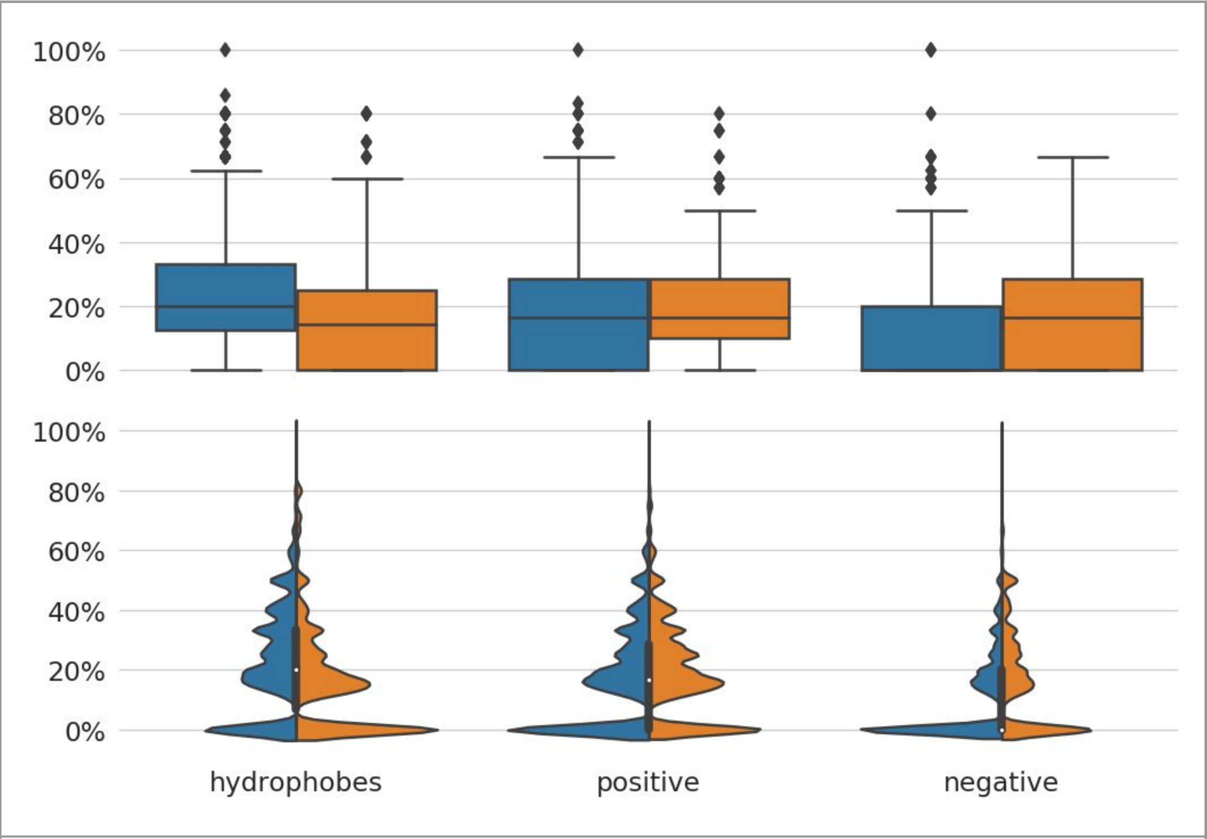


Figure 4.3.5 - The exposed neighbourhoods hydrophobicity and charge.
The first (upper) part shows the computed hydrophobicity and charge as a boxplot. The second (lower) part of this figure shows the corresponding violinplot to get an idea about the distributions. The peripheral protein dataset is represented in blue and the reference dataset is represented in orange.

Class / Combination / Condition	Periph. protein data neighbourhoods	Reference data neighbourhoods
Hydrophobicity ∩ (Positives > Negatives)	36.13%	28.35%
Hydrophobicity ∩ (Positives < Negatives)	17.18%	22.01%
Hydrophobicity ∩ (Positives == Negatives > 0)	10.59%	7.57%
Hydrophobicity ∩ (Positives == Negatives == 0)	12.90%	7.57%
Other (with no hydrophobicity)	23.20%	34.14%

Table 4.3.2 - Charge properties of exposed neighbourhoods with non-zero hydrophobicity
The table shows how frequent neighbourhoods with a non-zero hydrophobicity are charged or neutral. The table also contains the percentage of other neighbourhoods with no hydrophobicity.

Let us have a look at the hydrophobicity and charge of the exposed neighbourhoods. We compare the figure 4.3.5 with the figure 4.3.2 from the previous subchapter 4.3.1.

The hydrophobicity decreased. The violinplot on figure 4.3.5 shows that the peak at zero is the largest peak of the whole distribution for both datasets. Compared to the violinplot on figure 4.3.2, we can see that there are fewer hydrophobes in the exposed neighbourhood. This trend is also visible if we compare the boxplots of those figures. The hydrophobicity for whole neighbourhoods from the peripheral protein data is between ca. twenty and forty percent for fifty percent of the data (figure 4.3.2). We can see that the hydrophobicity for exposed neighbourhoods from the peripheral protein data got reduced with roughly ten percent points. Fifty percent of data lies between ten and thirty percent (figure 4.3.5). We can observe a similar decrease of the hydrophobicity for exposed neighbourhoods from the reference data. Here, the hydrophobicity of fifty percent of the data is roughly between ten and thirty percent for whole neighbourhoods (figure 4.3.2). It decreased, got more spread and lies between zero and twenty-five percent for exposed neighbourhoods (figure 4.3.5).

The spread of the positive charge increased very slightly for the exposed neighbourhood. We can clearly see on the violinplot from figure 4.3.5 that the peaks at zero for the positive charge are larger than the same peaks on the violinplot from figure 4.3.2. That indicates less positive charged neighbours in general. However, the upper quartile of the corresponding boxplot reaches almost thirty percent (figure 4.3.5) and the median also increased a little bit for the exposed neighbourhood. There are fewer positive charged neighbours in the exposed neighbourhood, but if a neighbourhood is positively charged, then the charge is slightly higher compared to the whole neighbourhood.

The distributions of the negative charge for both datasets changed too and the negative charge decreased a lot for exposed neighbourhoods from the peripheral data (figure 4.3.5). We can see a large peak at zero on the violinplot and also that the median is zero. The negative charge for exposed neighbourhoods from the reference data, however, changed not that much. Here, we can see some minor changes such as less total spread but higher spread for seventy-five percent of the data and a slightly higher median value.

Summarized, we can see that there are fewer hydrophobes and fewer charged amino acids in the exposed neighbourhood. Yet hydrophobes are still more present in exposed neighbourhoods from peripheral protein data than in exposed neighbourhoods from the reference data. However, we see less differences among properties between datasets on the exposed neighbourhood than on the overall neighbourhood.

Combining hydrophobicity with charge for the exposed neighbourhood results in the table 4.3.2. In exposed neighbourhoods with hydrophobicity greater than zero, the positive charge seems still to dominate. Compared to the table 4.3.1 from the previous subchapter, we can see that the hydrophobicity for exposed neighbourhoods decreased with roughly seven percent points for both datasets. We also see that hydrophobic, exposed and negative charged neighbourhoods decreased with six percent points for the peripheral protein data and five percent points for the reference data. The hydrophobic and neutral charged neighbourhoods with both positive charged and negative charged amino acids decreased too with about five percent points, but those with no positive charged nor negative charged amino acids increased with three percent points on both datasets.

We take with us that all twenty resides can be present in exposed neighbourhoods of hydrophobic protrusions as well. There was not so large differences to the overall neighbourhood from chapter 4.3.1 but arginine (ARG), glutamic acid (GLU) and lysine (LYS) are more frequent in the exposed neighbourhood. Alanine (ALA), cysteine (CYS) and valine (VAL) are less frequent compared to the overall neighbourhood. Cysteine (CYS) is not a very common neighbour in the exposed neighbourhood anymore and its presence decreased a lot for the peripheral protein dataset compared with other changes. The difference of cysteine (CYS) between datasets is not that big in the exposed neighbourhood as in the overall neighbourhood. On the other hand, not only the presence but also the difference of glutamic acid (GLU) between datasets increased.

Exposed neighbourhoods of hydrophobic protrusions have significantly fewer hydrophobes compared to whole neighbourhoods, but we can still see that hydrophobicity is dominating. We can also see that the exposed neighbourhood is less charged in general. Regardless of that we can still see that the positivity is dominating among exposed neighbourhoods with non-zero hydrophobicity. We can still see that hydrophobicity and the positive charge in a combination seems to be more distinct in exposed neighbourhoods of hydrophobic protrusions from protein peripheral data than in exposed neighbourhoods from the reference data.

4.3.3 Exposed neighbourhood of co-insertables vs. exposed neighbourhood of non-protrusion, exposed hydrophobes

Fuglebakk and Reuter (2018) showed that the definition of the likely inserted hydrophobe coincides with the binding sites of known peripheral proteins. They found first all hydrophobic protrusions that do have other hydrophobic protrusions in its close neighbourhood and they call those the co-insertables. Then the co-insertable with the highest number of other hydrophobic protrusions in its close neighbourhood and with the lowest local protein density becomes the proteins likely inserted hydrophobe.

We can make use of that knowledge and refine our previous analysis by restricting to co-insertables only. This increases the probability to study only neighbourhoods containing potential membrane binding amino acids. However, this leads to a problem with our datasets, because the co-insertable feature is separating the peripheral protein dataset from the reference dataset quite good, as shown by Fuglebakk and Reuter (2018). We find 836 co-insertables in the peripheral protein dataset, whereas we only find 53 co-insertables in the reference dataset. At this point, I decided to compare against the neighbourhood of non-protrusion but still exposed hydrophobes from the reference dataset instead, which are 11.936 neighbourhoods. This covers the reference dataset quite a lot of course. Here, we run into that we will find neighbourhoods which share common neighbour amino acids many times. This can result in more similar neighbourhoods, especially for co-insertables too because of its natural proximity to each other. I think this is important to have in mind later on when analysing the results.

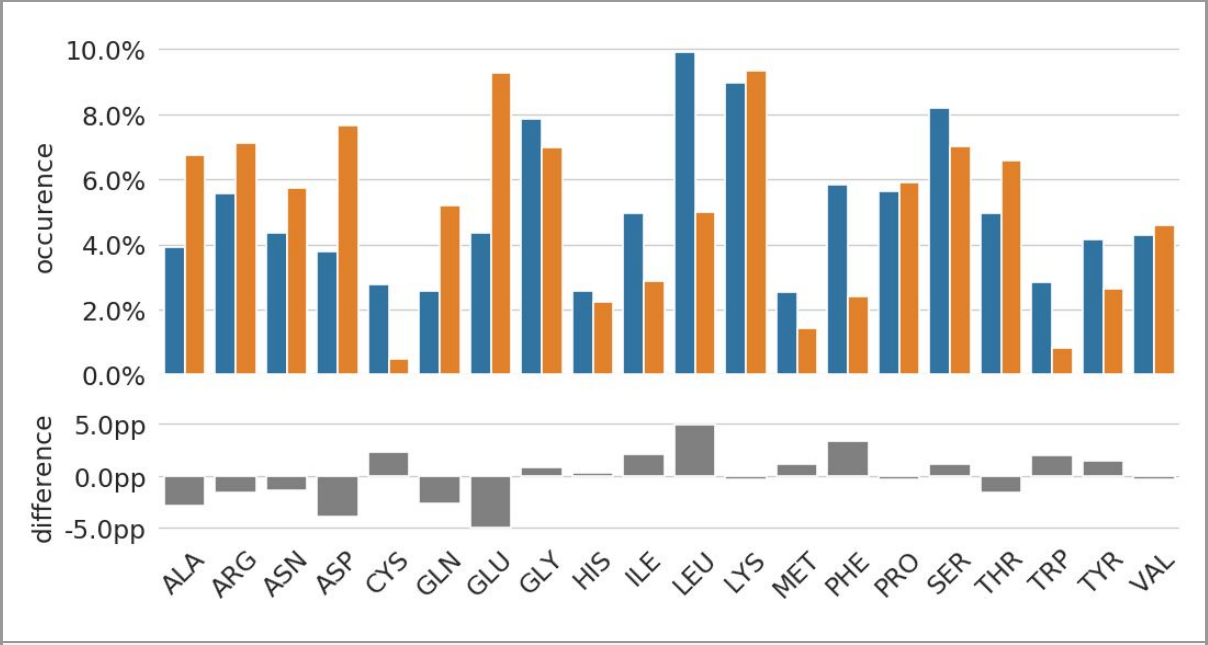


Figure 4.3.6 - The exposed neighbourhood of co-insertables vs. non-protrusion but exposed hydrophobes.
The first (upper) part of this figure shows the relative frequencies for both the exposed neighbourhood of co-insertables from the peripheral protein dataset (blue bars) and the exposed neighbourhood of non-protrusion but exposed hydrophobes from the reference dataset (orange bars). The second (lower) part of this figure shows the difference of the relative frequencies between the two datasets in percent points (pp). Here, a positive value means that there is a higher relative frequency for a particular amino acid in the peripheral protein dataset.

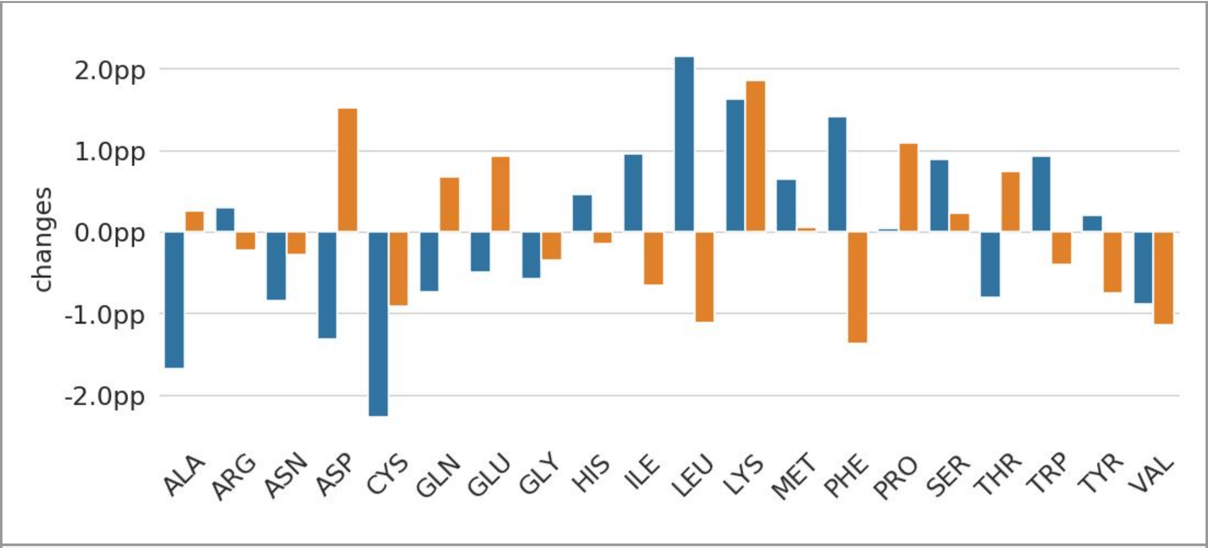


Figure 4.3.7 - Changes from the whole neighbourhood to the exposed neighbourhood of co-insertables and non-protrusion but exposed hydrophobes.
This figure shows the changes of the barplot from figure 4.3.1 (The neighbourhood of hydrophobic protrusions) to the figure 4.3.6 (The exposed neighbourhood of co-insertables & non-protrusion hydrophobes) in percent points (pp). Changes for the data from the peripheral protein dataset are represented as blue bars and changes for the data from the reference dataset are represented as orange bars.

We can observe from both the figure 4.3.6 and 4.3.7 that there are major differences compared to the overall neighbourhood from the section 4.3.1.

In the exposed neighbourhood of co-insertables from the peripheral protein dataset, leucine (LEU), lysine (LYS) and Serine (SER) are the three most common amino acids. Compared to the overall neighbourhood of all hydrophobic protrusions, all three amino acids became more common. Leucine (LEU) increased with over two percent points. On the other hand we do find significantly less cysteine (CYS) and alanine (ALA) in the exposed neighbourhood of co-insertables, which got reduced with over two percent points and one point five percent points respectively, compared to the overall neighbourhood. In the exposed neighbourhood of non-protrusion but still exposed hydrophobes from the reference dataset we find lysine (LYS), glutamic acid (GLU) and aspartic acid (ASP) to be the most common amino acids. Those amino acids became also more present compared to the overall neighbourhood of hydrophobic protrusions.

As the second part of the figure 4.3.6 shows, the exposed neighbourhood of co-insertables from the peripheral protein dataset differs from the exposed neighbourhood of non-protrusion but still exposed hydrophobes in the way that leucine (LEU), phenylalanine (PHE) and cysteine (CYS) are more present in the former, whereas glutamic acid (GLU) and aspartic acid (ASP) are more present in the latter. The difference of both leucine (LEU) and glutamic acid (GLU) is around five percent points between datasets.

Comparing the figure 4.3.7 to the figure 4.3.4 from the previous section 4.3.2 is quite interesting. The figure 4.3.4 shows the changes from the overall neighbourhood to the exposed neighbourhood of hydrophobic protrusions for both datasets. We saw that the changes were going in the same direction for both datasets, more or less. On the figure 4.3.7 however, we can observe that the changes are going more aggressively the opposite way, based on the datasets. If a type of amino acid increased on one dataset, we see that the same type of amino acid decreased for the other dataset in many cases. We do not see this for all types of amino acids though, some major exceptions are cysteine (CYS), lysine (LYS) and valine (VAL).

The figure 4.3.8 on the next page shows the hydrophobicity and charge for both the exposed neighbourhood for co-insertables from the peripheral protein dataset and the exposed neighbourhood for non-protrusion, exposed hydrophobes from the reference dataset. As having a hydrophobic protrusion in the close neighbourhood is part of the definition of a co-insertable, we see no co-insertable with a non-hydrophobic neighbourhood.

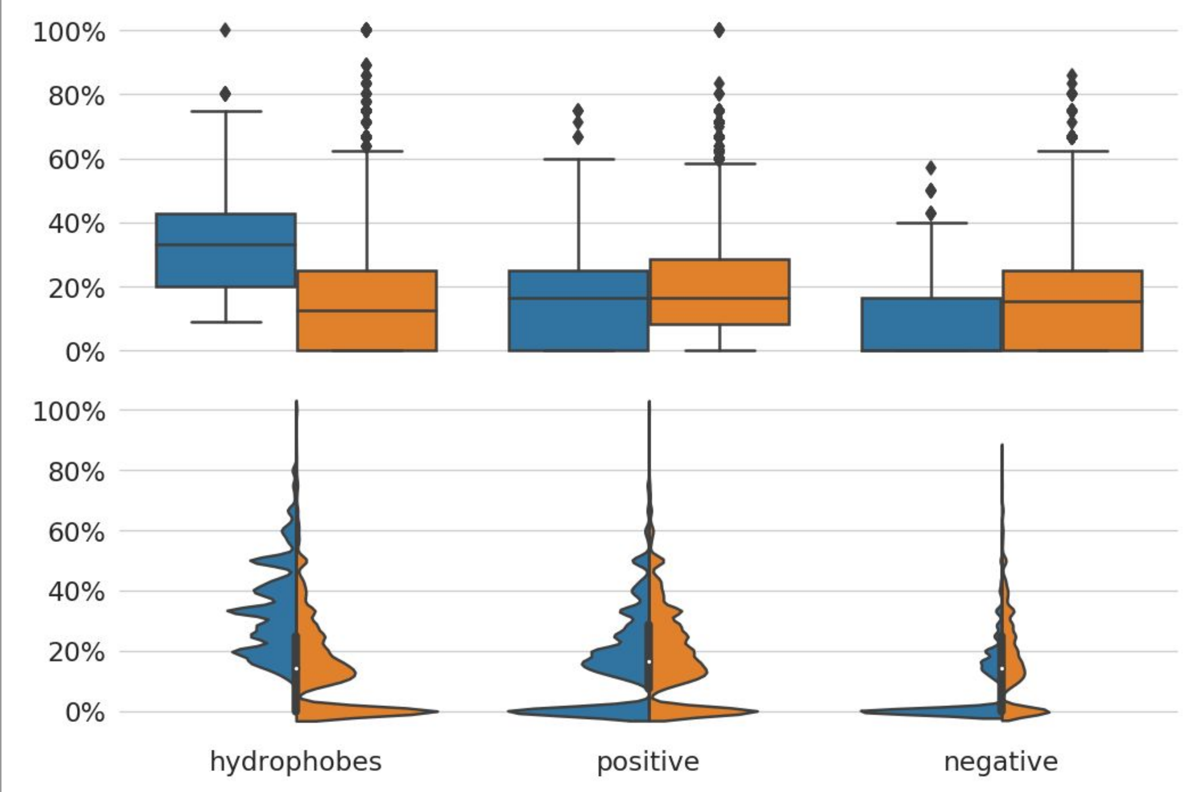


Figure 4.3.8 - Hydrophobicity and charge for the exposed neighbourhood of co-insertables and non-protrusion hydrophobes
The first (upper) part shows the computed hydrophobicity and charge as a boxplot. The second (lower) part of this figure shows the corresponding violinplot to get an idea about the distributions. The co-insertables from the peripheral protein dataset are represented in blue and the non-protrusion hydrophobes from the reference dataset are represented in orange.

Class / Combination / Condition	Periph. protein data neighbourhoods (co-insertables)	Reference data neighbourhoods (non-protr. hydr.)
Hydrophobicity ∩ (Positives > Negatives)	50.84%	25.45%
Hydrophobicity ∩ (Positives < Negatives)	17.58%	22.12%
Hydrophobicity ∩ (Positives == Negatives > 0)	13.64%	12.83%
Hydrophobicity ∩ (Positives == Negatives == 0)	17.94%	6.02%
Other (with no hydrophobicity)	0.00%	33.58%

Table 4.3.3 - Charge properties of exposed neighbourhoods of co-insertables and non-protrusion hydrophobes with non-zero hydrophobicity
The table shows how frequent neighbourhoods with a non-zero hydrophobicity are charged or neutral. The table also contains the percentage of other neighbourhoods with no hydrophobicity.

The figure 4.3.8 shows that the exposed neighbourhoods hydrophobicity stays roughly between ten and seventy-five percent for co-insertables from the peripheral protein dataset. Here, the common relative number of hydrophobic neighbours is between twenty and forty percent as shown by the corresponding boxplot. Compared to the figure 4.3.2 which shows the same properties for the overall neighbourhood of hydrophobic protrusions and to the figure 4.3.5 which shows the same properties for the exposed neighbourhood of hydrophobic protrusions, the hydrophobicity increased for the exposed neighbourhood of co-insertables.

We see less hydrophobic neighbours in the exposed neighbourhood of non-protrusion but exposed hydrophobes from the reference dataset. Roughly seventy-five percent of those exposed neighbourhoods contain not more than twenty percent hydrophobic amino acids. We observe from the corresponding violinplot that there are again a lot neighbourhoods which does not contain any hydrophobic amino acid.

Comparing the figure 4.3.8 to the figures 4.3.5 from the previous section about the exposed neighbourhood of hydrophobic protrusions, we can see some spread variation and more outsiders for the positively charged amino acids and the negatively charged amino acids within the exposed neighbourhood. Overall the violinplot for the positively charge of neighbourhoods did not change a lot. But we can see that the exposed neighbourhood of non-protrusion but exposed hydrophobes from the reference dataset are more negative charged than the exposed neighbourhood of hydrophobic protrusions from the reference dataset. Here, notice that peak at zero on the corresponding violinplot is smaller on the figure 4.3.8 than on the figure 4.3.5 compared to the rest of the distribution.

Having a look at the table 4.3.3, the neighbourhoods hydrophobicity in combination with the neighbourhoods positive charge is dominating for the exposed neighbourhoods of co-insertables. Here, over fifty percent of the those neighbourhoods are both hydrophobic and positive charged. This is not the case for the neighbourhood of non-protrusion but still exposed hydrophobes, where this feature combination only stays at about twenty-five percent. Combining hydrophobicity and the negative charge gives almost the same results as from the previous sections table 4.3.2 for the neighbourhoods of interest from both datasets. However, we see that the hydrophobic but neutral charged neighbourhoods increased.

We can take with us from this section, that not only the neighbourhoods hydrophobicity but also the neighbourhoods positive charge can play a role in the exposed neighbourhood of co-insertable from the peripheral protein dataset. We see also that we do not find this feature combination with such a dominance for the exposed neighbourhood of non-protrusion but still exposed hydrophobes from the reference dataset. While the combination of being hydrophobe and positively charged increased for the exposed neighbourhood of co-insertables from the peripheral protein dataset, the overall negatively charge for non-protrusion, exposed hydrophobes from the reference dataset increased too. Otherwise we see still all twenty amino acids present and some major frequency changes compared to the previous result sections. This time, both leucine (LEU) and glutamic acid (GLU) are the amino acids with the largest differences in percent points between datasets. Both amino acids were already mentioned in the previous sections and stay interesting for further studies.

Chapter 5

Discussion

Chapter 5.1

Discussing the results

The first major problem of my master thesis was about to program a useful tool that is visualizing Fuglebakk and Reuters (2018) protrusion model. I am able to present my convex hull plugin for the popular molecular viewer UCSF Chimera (Pettersen *et al.*, 2004) as a result and solution for this problem. The plugin, which is described in more details in chapter 4.2, is easy to install in general and easy to use for everyone who is familiar with the UCSF Chimera software. The plugin computes a convex hull and shows all protrusions for a protein of interest. It also gives the possibility to play with variables of Fuglebakk and Reuters (2018) protrusion model, showing the computed results instantly on the protein structure in realtime to the user.

The result in form of a plugin means that it now is significantly easier and more interactive to apply the protrusion model. Even non-informaticians can now visualize and study Fuglebakk and Reuters (2018) protrusion model, without the need to program the protrusion definition or a visualization themselves. Said that, it is obvious that the result matters, because the plugin offers an enormous time advantage in finding protrusions on a protein of interest compared to the need to first create a custom implementation previously. As the plugin also gives the possibility to change variables of the protrusion definition and change the look & feel, it is now easier to do a custom visual analysis and good looking screenshots as well.

But there are also important limitations to mention. The plugin can just show the proteins convex hull, protrusions and hydrophobic protrusions. However, the user can not use the plugin to highlight hydrophobic co-insertables or the likely inserted hydrophobe. The user can archive this by investigating hydrophobic protrusions the plugin found, using some of the built-in functions and tools of UCSF Chimera, and applying all required and missing definition themselves. But this is a very advanced use of the software, which maybe not everyone is willing or able to do. This limitation is the reason why I named it for the convex hull plugin, because the name indicates what the plugin mainly does and what the user can expect. It computes convex hulls and highlights amino acids on the convex hull with a low local protein density. As mentioned already in section 4.2.2, there is also a performance limitation. The plugin computes and shows the user's input in realtime. If the protein complex of interest is a very large one, then the computation time increases and the plugin may not respond in realtime anymore; showing a loading screen. However, and fortunately, this is not the case for most peripheral proteins, which the user can compute without any performance issues on low spec laptops. Another limitation may also be the license of the plugin and UCSF Chimera duo to the need of being used non-profit or part of academic work.

The second major aim of my master thesis was to characterize the environment of hydrophobic protrusions, finding amino acids patterns at membrane-binding interfaces of peripheral proteins. I calculated the amino acid composition around hydrophobic protrusions and whether amino acids with electrostatic properties could be found there. I can present as a result that I have introduced some new definitions such as *the neighbourhood, the neighbourhoods hydrophobicity* and *the neighbourhoods charge*, which I of course applied to the datasets from Fuglebakk and Reuter (2018). Later on I also refined those definitions to only be applied to surface-exposed amino acids, defining the *exposed neighbourhood* and excluding amino acids which may be positioned too deep into the protein structure.

The analysis shown in chapter 4.3 suggests that all twenty amino acids are more or less present in neighbourhoods of hydrophobic protrusions and that no particular amino acid is very dominating. We also got an impression about which amino acids differ the most between the datasets of peripheral proteins and the reference dataset. It is important to notice that the amino acid frequencies shown at figure 4.3.1, 4.3.3 and 4.3.6 are only representing a mean occurrence of amino acids found in all neighbourhoods. The results do not reveal a clear pattern or model for amino acid frequencies in the neighbourhood of hydrophobic protrusions, which would cause the datasets to differ a lot. But we can see some interesting trends developing from the overall neighbourhood to the exposed neighbourhood of co-insertables from the peripheral protein dataset, as *e.g.* the increase of leucine (LEU) or the decrease of cysteine (CYS). This results matters, because we now know that we have to look at neighbourhoods more individually and from a more functional point of view; seeking for biochemical or biophysical properties.

If we consider neighbourhoods individually and if we look for the neighbourhoods hydrophobicity and charge, then the results demonstrate clearly that hydrophobic neighbourhoods of hydrophobic protrusions from the peripheral protein dataset contain more positively charged amino acids than hydrophobic neighbourhoods from the reference dataset. The combination of hydrophobic and positively charged amino acids in a neighbourhood becomes even more dominating for exposed neighbourhoods of co-insertables compared to exposed neighbourhoods of non-protrusion hydrophobes. This is an interesting pattern we found, but the finding goes in a direction that was expected due to the previous biological knowledge. Known binding sites of membrane-binding peripheral proteins are mainly made of hydrophobic and positively charged amino acids (Johnson and Cornell, 1999; Luckey, 2014) and the results clearly tend to fulfill that.

The results matter in the sense that we indeed can use a computational model to approach the study of potential membrane-binding sites of peripheral proteins successfully, based on the model from Fuglebakk and Reuter (2018). We see that their model can be improved if exposed, positively charged neighbours are also taken into account. My neighbourhood definition played an important role to characterize the environment around hydrophobic protrusions and my definition could also be used to search for other biochemical and biophysical features in the future. Said that, it is also important to notice some limitations. The maybe most important limitation to mention is that I only looked into the neighbourhood of co-insertables. One may want to extend my work by also looking at neighbourhoods of the likely inserted hydrophobes. My neighbourhood definition is dependent on Fuglebakk and Reuters (2018) distance criteria, which also is an important limitation. Another limitation is the data. The datasets are quite small, which made it challenging to compare features and find differences. The peripheral protein dataset does not contain any information about the real, experimentally verified binding sites, which makes the development of a good predictive model difficult.

Chapter 5.2

What practical actions or scientific studies should follow?

Thinking more biology at this point, no hydrophobicity in the neighbourhood may result in more unstable, lonely hydrophobes that are not likely to be membrane-binding. Fuglebakk and Reuters (2018) definition of co-insertables ensures that there exists at least one hydrophobic neighbour in the close neighbourhood. Also hydrophobes which do have a negatively charged neighbourhood may not be membrane-binding either, because of the anionic lipids in the membrane. Excluding those co-insertables with negatively charged neighbourhoods and focusing on co-insertables with a positively charged or neutral neighbourhood can give interesting adjustments to the study from Fuglebakk and Reuter (2018) for further studies. It would be interesting to see if co-insertables with a positively charged neighbourhood coincides more often with known binding sites. This leads me also to the idea of “super co-insertables”, which do have both hydrophobic and positively charged amino acids in their neighbourhoods. Further studies could define such super co-insertables more in detail and show if and how much they would improve the model.

Another study I can think of would be about the combination of different protrusions models. Fuglebakk and Reuter (2018) defined hydrophobic protrusions in their model. It would be interesting if we also look at *e.g.* positively charged protrusions. In the positive model, also the convex hull may become computed just among positives. We can then combine the models and look if some protrusions are near each other and look if this would coincide with known binding sites.

It would be interesting to enrich the dataset if possible. The dataset could be enriched with sequence data, having the possibility to understand where in the peptide-chain of a peripheral protein potential binding sites exists. This may also increase the understanding of protein folding of peripheral proteins. If the folding of a binding site already can be predicted based on sequence data, then the computational model would become very strong I think.

A very smart thought may be to use the current dataset from a new perspective. If we know the binding sites of a peripheral protein, then the rest of the surface could be treated as a non-binding reference surface to compare to.

I implemented a plugin for the popular molecular viewer UCSF Chimera. But the plugin is still not perfect and does have some missing features such as the ability to highlight the most likely hydrophobe. A practical action would be to continue developing on my plugin. There are also other popular molecular viewers out there. Programming tools that bring the protrusion model from Fuglebakk and Reuter (2018) to life would also be interesting programming projects. For example a webtool would be very interesting, such that everyone can visualize convex hulls and protrusions in their web browser.

Chapter 5.3

Why was machine learning not yet an appropriate solution?

When you read the analysis part of my master thesis, you may wonder why I did not approach the datasets with a machine learning method. Predicting parts of the data, especially protrusions, amino acid compositions or binding sites based on neighbourhoods, could be very interesting to do. We also see that machine learning methods do have some sort of a hype these days, because they can be very nice, high level predictors. For the following paragraphs, I used Mitchell (1997), Zvelebil and Baum (2008) and Luckey (2014) more freely as my references.

I think that considering machine learning methods in data analysis should always be done, as there are a lot of advantages if you can describe your data with good, predictive models. But machine learning methods are not always the appropriate solution. Sometimes, a simple frequency analysis or other traditional, statistical analysis can give us more knowledge than a bad performing machine learning model.

When you want to do machine learning, then you need data. You need a lot of data. Most machine learning algorithms may predict quite well on just a little data, but then the correctness of the model must always be criticised. You may run into problems such as overfitting your data, which means that you perform quite good on your data but very bad on unseen data. If your dataset is too small, it is very complicated to show how your model performs to unseen data, as you may use most of your data to train your model. You also encounter problems using common learning methods *e.g.* using cross validation the proper way, because splitting a too small dataset can give you unwanted effects.

Other problems you may have are the biological nature of data. Biological data can be very noisy. The reason for noise can be complex, either it is the noisy nature itself, the noisy way the data was collected or both. If you have too little data, the risk of making a model that predicts just the noise is very high. The quality of the data is therefore very important to consider as well when doing a machine learning approach.

The datasets from Fuglebakk and Reuter (2018) are quite small compared to other big data used in machine learning approaches. We also encounter the problem that Fuglebakk and Reuter (2018) already separated the peripheral protein dataset from the reference dataset with their definition of hydrophobic protrusions and co-insertables relatively good. When we now begin to analyse the data more, I noticed that we are just analysing an even smaller subset of the data in most of the cases. A machine learning approach is therefore not appropriate as the original dataset already was small and even got reduced too much. The risk of overfitting is too high and the model may predict too bad in general, maybe a model would just predict noise.

It is also important to notice that we want to study membrane binding of peripheral proteins. I do not see why predicting protrusions or neighbourhoods in general could help us to improve the understanding at this point of the study, at least not with so little data. Focusing on biochemical or biophysical properties, extending the already existing and working model from Fuglebakk and Reuter (2018) seems more appropriate at the current state of the art.

Chapter 6

Bibliography & References

24.1. Tkinter — Python interface to Tcl/Tk — Python 2.7.18 documentation (no date). Available at: https://docs.python.org/2.7/library/tkinter.html (Accessed: 15 May 2020).

Arnold, K., Gosling, J. and Holmes, D. (2012) *The Java Programming Language*. Prentice Hall.

Atomic Coordinate Entry Format Version 3.3 (no date). Available at: https://www.wwpdb.org/documentation/file-format-content/format33/v3.3.html (Accessed: 14 May 2020).

Barber, C. B. *et al.* (1996) 'The quickhull algorithm for convex hulls', *ACM Transactions on Mathematical Software (TOMS)*, pp. 469–483. doi: 10.1145/235815.235821.

Berman, H. M. *et al.* (2000) 'The Protein Data Bank', *Nucleic acids research*, 28(1), pp. 235–242.

Chen, H. and Zhou, H.-X. (2005) 'Prediction of solvent accessibility and sites of deleterious mutations from protein sequence', *Nucleic acids research*, 33(10), pp. 3193–3199.

Chimera Programmer's Guide (no date). Available at: https://www.cgl.ucsf.edu/chimera/docs/ProgrammersGuide/index.html (Accessed: 15 May 2020).

Costa-Luis, C. O. da and da Costa-Luis, C. O. (2019) 'tqdm: A Fast, Extensible Progress Meter for Python and CLI', *Journal of Open Source Software*, p. 1277. doi: 10.21105/joss.01277.

Fuglebakk, E. and Reuter, N. (2018) 'A model for hydrophobic protrusions on peripheral membrane proteins', *PLOS Computational Biology*, p. e1006325. doi: 10.1371/journal.pcbi.1006325.

Git (no date). Available at: https://git-scm.com/ (Accessed: 8 May 2020).

Grauffel, C. *et al.* (2013) 'Cation−π Interactions As Lipid-Specific Anchors for Phosphatidylinositol-Specific Phospholipase C', *Journal of the American Chemical Society*, pp. 5740–5750. doi: 10.1021/ja312656v.

Hinsen K. (2000), The molecular modeling toolkit: A new approach to molecular simulations. J Comput Chem. 2000; 21(2):79–85. https://doi.org/10.1002/(SICI)1096-987X(20000130)21:2%3C79::AID-JCC1%3E3.0.CO;2-B

Humphrey, W., Dalke, A. and Schulten, K. (1996) 'VMD: Visual molecular dynamics', *Journal of Molecular Graphics*, pp. 33–38. doi: 10.1016/0263-7855(96)00018-5.

Hunter, J. D. (2007) 'Matplotlib: A 2D Graphics Environment', *Computing in Science & Engineering*, pp. 90–95. doi: 10.1109/mcse.2007.55.

jinxuan (no date) *jinxuan/convexhull*, *GitHub*. Available at: https://github.com/jinxuan/convexhull (Accessed: 9 May 2020).

Johnson, J. E. and Cornell, R. B. (1999) 'Amphitropic Proteins: Regulation by Reversible Membrane Interactions (Review)', *Molecular membrane biology*. Mol Membr Biol, 16(3). doi: 10.1080/096876899294544.

Kan, C. N. E. (2018) 'Data Science 101: Is Python better than R? - Towards Data Science'. Towards Data Science. Available at: https://towardsdatascience.com/data-science-101-is-python-better-than-r-b8f258f57b0f (Accessed: 7 May 2020).

Lamy, J.-B. *et al.* (2016) *Figure 2. Venn diagram showing 9 properties of the 20 amino-acids*, *ResearchGate*. Available at: https://www.researchgate.net/figure/Venn-diagram-showing-9-properties-of-the-20-amino-acids_fig2_307573998 (Accessed: 2 June 2020).

Libretexts (2013) 'Lipid Bilayer Membranes'. Libretexts. Available at: https://chem.libretexts.org/Bookshelves/Biological_Chemistry/Supplemental_Modules_(Biological_Chemistry)/Lipids/Applications_of_Lipids/Lipid_Bilayer_Membranes (Accessed: 8 June 2020).

Lomize, M. A. *et al.* (2012) 'OPM database and PPM web server: resources for positioning of proteins in membranes', *Nucleic Acids Research*, pp. D370–D376. doi: 10.1093/nar/gkr703.

Luckey, M. (2014) *Membrane Structural Biology: With Biochemical and Biophysical Foundations*. Cambridge University Press.

McKinney, W. (2010) 'Data Structures for Statistical Computing in Python', *Proceedings of the 9th Python in Science Conference*. doi: 10.25080/majora-92bf1922-00a.

Miller, S. *et al.* (1987) 'Interior and surface of monomeric proteins', *Journal of molecular biology*, 196(3), pp. 641–656.

Millman, K. J., Jarrod Millman, K. and Aivazis, M. (2011) 'Python for Scientists and Engineers', *Computing in Science & Engineering*, pp. 9–12. doi: 10.1109/mcse.2011.36.

Mitchell, T. M. (1997) *Machine Learning*. The McGraw-Hill Companies, Inc.

Mitternacht, S. (2016) 'FreeSASA: An open source C library for solvent accessible surface area calculations', *F1000Research*, 5, p. 189.

Oberoi, A. and Chauhan, R. (2019) 'Visualizing data using Matplotlib and Seaborn libraries in Python for data science', *International Journal of Scientific and Research Publications (IJSRP)*, p. 8733. doi: 10.29322/ijsrp.9.03.2019.p8733.

Oliphant, T. (2015) *Guide to NumPy: 2nd Edition*. CreateSpace.

OPM (no date). Available at: https://opm.phar.umich.edu/ (Accessed: 14 May 2020).

PDB101: Learn: Guide to Understanding PDB Data: Methods for Determining Structure (no date) *RCSB: PDB-101*. Available at: https://pdb101.rcsb.org/learn/guide-to-understanding-pdb-data/methods-for-determining-structure (Accessed: 13 May 2020).

Pettersen, E. F. *et al.* (2004) 'UCSF Chimera--a visualization system for exploratory research and analysis', *Journal of computational chemistry*, 25(13), pp. 1605–1612.

Project Jupyter (no date). Available at: https://www.jupyter.org (Accessed: 6 May 2020).

'PyCharm' (no date). Available at: https://www.jetbrains.com/de-de/pycharm/ (Accessed: 8 May 2020).

Raschka, Sebastian (2017). Biopandas: Working with molecular structures in pandas dataframes. The Journal of Open Source Software, 2(14), jun 2017. doi: 10.21105/joss.00279. URL http://dx.doi.org/10.21105/joss.00279

Rcpp · Advanced R (no date). Available at: http://adv-r.had.co.nz/Rcpp.html (Accessed: 7 May 2020).

Reuter Group – CBU (no date). Available at: https://www.cbu.uib.no/reuter/ (Accessed: 12 May 2020).

Rose, G. D. *et al.* (1985) 'Hydrophobicity of amino acid residues in globular proteins', *Science*, 229(4716), pp. 834–838.

RStudio | Open source & professional software for data science teams (no date). Available at: https://rstudio.com/ (Accessed: 7 May 2020).

R: The R Project for Statistical Computing (no date). Available at: https://www.r-project.org/ (Accessed: 7 May 2020).

The World's Most Popular Data Science Platform | Anaconda (no date) *Anaconda*. Available at: https://www.anaconda.com/ (Accessed: 6 May 2020).

Tien, M. Z. *et al.* (2013) 'Maximum allowed solvent accessibilites of residues in proteins', *PloS one*, 8(11), p. e80635.

UCSF Chimera Non-Commercial Software License Agreement (no date). Available at: https://www.cgl.ucsf.edu/chimera/license.html (Accessed: 23 May 2020).

Van Rossum, G. & Drake Jr, F.L., 1995. Python reference manual, Centrum voor Wiskunde en Informatica Amsterdam.

Watson, J. D., Baker, T. A. and Bell, S. P. (2014) *Molecular Biology of the Gene*. Benjamin-Cummings Publishing Company.

Wimley, W. C. and White, S. H. (1996) ‘Experimentally determined hydrophobicity scale for proteins at membrane interfaces’, *Nature structural biology*, 3(10), pp. 842–848.

Zvelebil, M. J. and Baum, J. O. (2008) *Understanding Bioinformatics*. Garland Science.