Investigations on Key Principles of PTP1B Selectivity

Dissertation zur Erlangung des akademischen Grades
des Doktors der Naturwissenschaften (Dr. rer. nat.)

eingereicht im Fachbereich Biologie, Chemie, Pharmazie
der Freien Universität Berlin

vorgelegt von

Alexandra Vanessa Olympia Naß

Berlin

2018

Die vorliegende Arbeit wurde von Oktober 2013 bis April 2018 unter der Leitung von Prof. Dr. Gerhard Wolber am Institut für Pharmazie der Freien Universität Berlin angefertig.

1. Gutachter: Prof. Dr. Gerhard Wolber
2. Gutachter: Prof. Dr. Peter Kolb

Disputation am 08. Juni 2018

Acknowledgements

First of all I want to express sincere gratitute to my supervisor Prof. Dr. Gerhard Wolber. Without his relentless scientific advice this thesis would not have been possible.
I am especially grateful for the freedom I enjoyed in his group and the happy and open-minded working athmosphere.

This pleasant environment relies heavily on all people involved. I therefore would also like to thank all my former and present co-workers in the AGWolber, especially Robert Schulz for being a smart-ass and David Schaller for an awesome Moonlight Session at the GCC.

I would also like to thank Prof. Dr. Peter Kolb for agreeing to co-examine this thesis.

Further I would like to thank the supervisors of the EUROPIN PhD program for their critical discussions on my work. Here I want to mention Prof. Dr. Wolfgang Sippl and Prof. Dr. Gerhard Ecker who enabled my first steps in Computational Drug Design.

I am grateful to Prof. Dr. Rosanna Maccari for the fruitful collaboration on PTP1B inhibitors and other topics.

Thanks also goes to Dr. Annette Kietzmann for her kind support during teaching.

Gratefully acknowledged is the computing cluster Soroban of the FU Berlin and the help of their support team.

Finally, I want to thank my family for their uncompromising continuous support.

Man sollte sich immer zu Ende wundern.

unbekannt

Contents

1. Introduction

1.1. Phosphatases

With about 30% of proteins being able to get phosphorylated, reversible phosphorylation is one of the major posttranslational modification mechanisms [1]. The phosphorylation state of a protein is determined by protein kinases, attaching phosphate groups to proteins, and protein phosphatases, catalyzing the reverse reaction [2]. While the important role of kinases has been established long ago, phosphatases were erroneously considered second row enzymes for maintaining a kinase dependent equilibrium until recently, but are now known to play critical and highly specific roles in many signaling processes such as growth, proliferation and metabolism [3, 4]. Furthermore have they been shown to act as positive as well as negative modulators in signaling [5]. Recent literature suggests that kinases are involved in controlling the amplitude of a signaling response, whereas phosphatases are controlling rate and duration of a response [6].

Originally, phosphatases were classified into Ser/Thr-specific, Tyr-specific and dual-specific phosphatases based on their substrate specificity [7]. However, newer studies do not support this differentiation based on substrate specificity, because many phosphatases show a broader range of accepted substrates than expected. Therefore Sacco et al. suggested a classification based on amino acid sequence similarity of the catalytic sites with Ser/Thr-specific and Tyr-specific phosphatases being further divided into different subgroups and dual-specific phosphatases classified in one family together with Tyr-specific phosphatases [7, 8].

Later investigations of phosphatase substrate selectivity interestingly revealed that they often do not show significant selectivity *in vitro*, but clear preference to phosphorylate certain substrates *in vivo*. Whereas one part of this *in vivo* selectivity can be related to non-catalytic phosphatase domains, regulating their activity or enriching substrate concentration in the environment of the phosphatase by targeting it to a certain compartment of the cell, there is still a considerable part of the selectivity that seems to be related to the catalytic domains of the phosphatases. This indicates that active site directed selective inhibition should be possible, but might be dependend on assay conditions [8].

Although protein-tyrosine phosphorylation only constitutes less than 1% of protein phosphorylation activity [1], protein tyrosine phosphatases are encoded by the largest family of phosphatase genes [9], which depicts their importance in phosphorylation mediated signaling. Protein tyrosine phosphatases share a so-called *signature motif*, which is the conserved sequence (H/V)C(X)$_5$R(S/T) in the active site [6]. This motif includes the cysteine working as the nucleophile of the catalytic substrate reaction as published by Pannifer et al. (Figure 1.1) [6, 10].

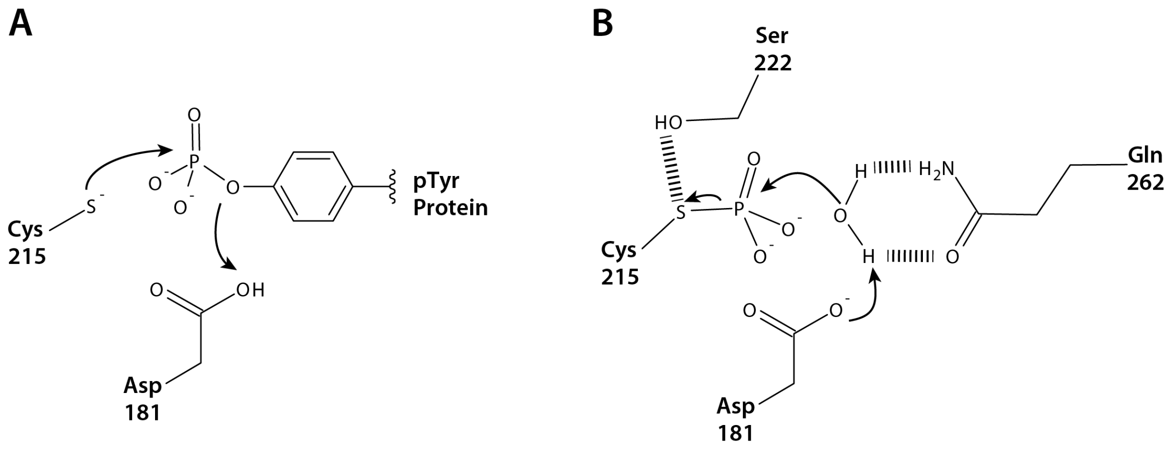


Figure 1.1.: Mechanism of protein-tyrosine phosphorylation. A: Formation of cysteinyl-phosphate, B: regeneration by hydrolysis. Adapted from Pannifer et al. [10].

1.2. Inhibition of Protein Tyrosine Phosphatase 1B

The most prominent member of the PTP superfamily, PTP1B, was purified over 25 years ago from human placenta [11]. It consists of 435 amino acids with residues 30-278 summarized as the catalytic domain and 35 C-terminal residues responsible for targeting the enzyme to the cytosolic face of the endoplasmic reticulum [12]. During many years of intense studies PTP1B has been validated as a drug target for diabetes and obesity as well as a promising target for different types of cancer [13, 14, 4]. PTP1B negatively modulates insulin and leptin signaling [5] and is overexpressed in the mentioned diseases [4]. Figure 1.2 depicts cellular pathways with PTP1B interference.

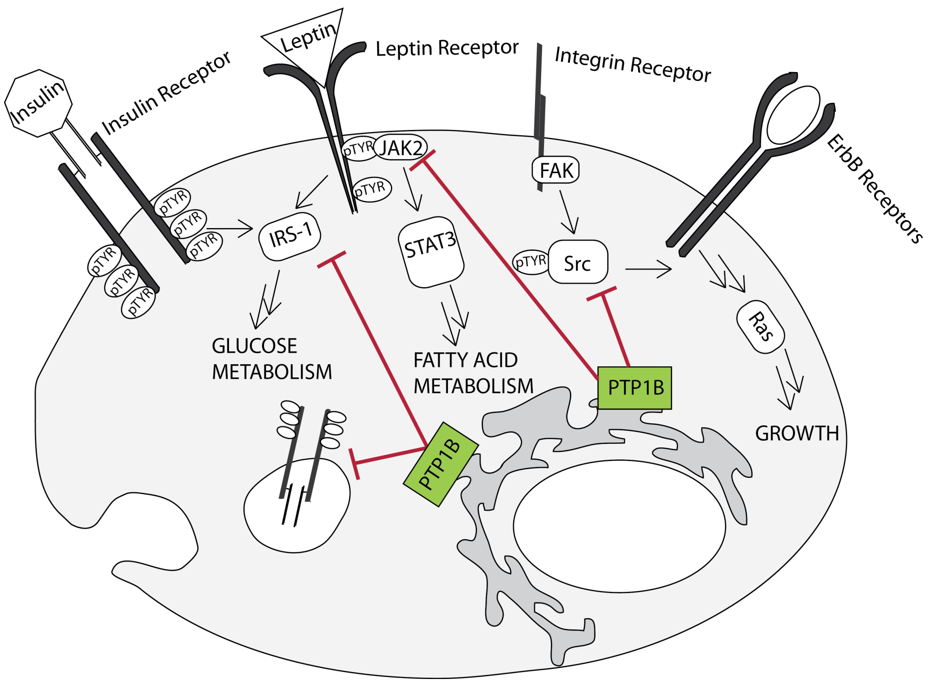


Figure 1.2.: PTP1B interfering with insulin, leptin and growth factor signaling. Adapted from Johnson et al. [15].

PTP1B knockout mice show enhanced insulin sensitivity, low postprandial serum glucose and insulin levels as well as resistance to obesity under a high-fat diet. Furthermore they seem otherwise healthy with no increased risk of cancer [16, 17]. Therefore the development of potent drugs promises increased insulin sensitivity without the weight gain, a side effect occurring with marketed insulin sensitizing drugs.

However, developing PTP1B modulators as drugs has been hampered by several challenges: One of them is the problem of bioavailability. Since the active site has developed to bind highly polar and charged phosphate substrates, tightly binding inhibitors show the same properties which are connected to low membrane permeability and therefore low ability to reach PTP1B localized at the endoplasmic reticulum *in vivo* [5]. Beneath prodrug modifications, it was recently proposed to use a special property of the protein to circumvent this issue: Due to its molecular environment, the catalytic cysteine of PTP1B shows a low pK_a of around 4.6 [2], which on the one hand enhances its nucleophilic properties for catalysis of the dephosphorylation, but on the other hand makes it prone to oxidation. After oxidation to sulfenic acid the reaction proceeds producing a 5-membered sulfenamide, connecting Cys215 and Ser216 in the catalytic site [4]. This cyclization leads to a conformational change protecting the protein from irreversible oxidation and facilitating reactivation by reduction. The resulting protein cavity, however, is less polar than the reduced version, but more open, and was suggested as a promising target structure for less polar inhibitors [3].

Unfortunately, the circumstances and the amount of oxidation *in vivo* are not thoroughly discovered. Therefore it remains unclear how relevant targeting this state of the protein might be for the treatment of the abovementioned diseases. Noteworthily, this susceptibility to oxidation has caused problems in high-throughput screening, often including oxidizing or peroxide releasing compounds, which is another obstacle in inhibitor design for PTP1B [4].

The most challenging part in the development of PTP1B targeting drugs, however, is related to the high degree of structural conservation throughout the active sites of PTPs [5]: An especially close relative of PTP1B –TC-PTP –shows an overall identity of 74% in the catalytic domain shared by both proteins and 100% sequence identity of catalytic site residues (T177-P185, H214-R221, Q266; PTP1B naming). A study by You-Ten et al.[18] led to the result that TC-PTP knockout mice die within 5 weeks after birth showing severe defects in T-Cell and B-Cell function. This is supported by the genetic association of the TC-PTP gene with inflammation and autoimmunity [19]. Therefore selectivity of PTP1B inhibitors against the highly similar TC-PTP seems strongly advised to prevent severe side effects in humans. While selectivity was discovered to be achievable over other PTPs, only few PTP1B inhibitors could be developed to at most moderate selectivity over TC-PTP [6].

Moreover, complicating structure-based drug design approaches, the flexible WPD (Trp, Pro, Asp) loop closes upon substrate binding, enabling catalytic activity of the enzyme. Furthermore, stronger inhibitors seem to bind to the closed conformation of the active site [15]. While for PTP1B both conformations were resolved in several crystal structures, there is only one crystal structure publicly available for TC-PTP [20]. This structure, however, depicts the open WPD loop conformation and therefore the less relevant structure for inhibitor design. Additionally, this structure is of low quality as discussed in Section 4.1.3 which restricts its use in detailed structure comparisons.

Despite the mentioned challenges, some progress has been made in the development of PTP1B inhibitors which is summarized in several reviews [15, 21, 12]. Major breakthroughs include: A) Discovery of the difluoromethylene phosphonate group as phosphotyrosine mimetic, which converted peptidic substrates to inhibitors [22], B) Identification of a second phosphotyrosine binding site (B-site) close to the catalytic site with slight differences in amino acid composition in TC-PTP compared to PTP1B

[23], C) Identification of early bi-pTYR-mimetic peptides not binding to the second phosphotyrosine binding site in crystal structures, but showing interactions with Arg47 (C-site), surprisingly still leading to moderate selectivity - about tenfold - against TC-PTP [24].

Later efforts concentrated on reducing the peptidic character of the inhibitors and reducing the charge, keeping the bidentate approach to increase selectivity [25, 26, 27, 28, 29]. This lead to carboxylic acid based inhibitors and finally the highly active thiadiazolidinone and isothiazolidinone derivatives which are stabilized in the active site by a hydrogen bonding network similar to that of pTYR residues [30, 31, 32, 12]. Figure 1.3 shows the binding site interactions of an thiadiazolidinone-derivative in comparison to the interactions of a phenylphosphate ligand. The thiadiazolidinone-derivative is able to effectively replace almost all hydrogen bonding interactions observed for the phenylphosphate moiety even replacing the active site water molecule and its mediated interactions. Newer studies state higher selectivity with selectivity ratios up to 45 against TC-PTP, however they lack detailed biological data like inhibition curves or data from kinetic analyses as well as structural proof of binding mode in form of crystal structures [33].

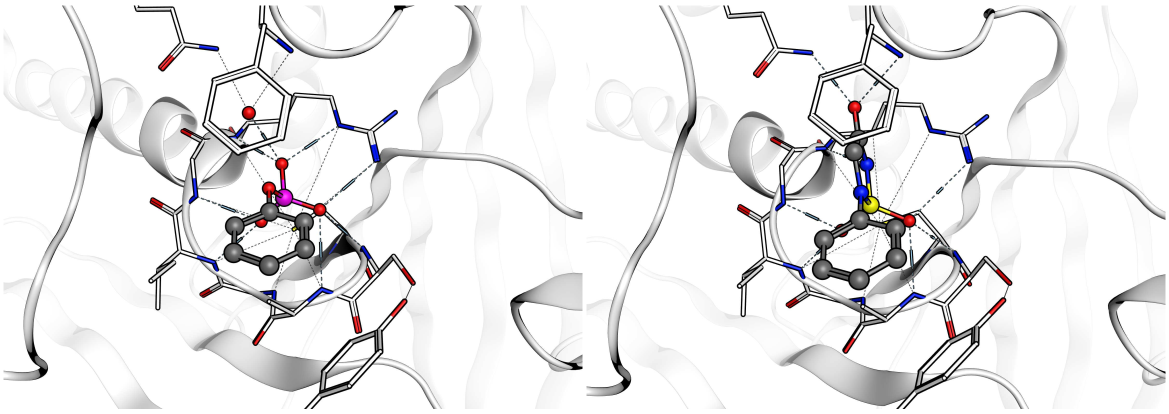


Figure 1.3.: Left: Phenylphosphate ligand and water molecule in the active site cavity of PTP1B (derived from PDB structure 1PTY); Right: Isothiadiazolidinone-derivative in the active site cavity of PTP1B (PDB structure 2GBE); light blue dashed lines represent hydrogen bonds.

2. Aim and Objectives

This study focuses on structure based design of small, active site, reversible selective inhibitors of PTP1B. The moderate selectivity of known substrates and inhibitors lead us to assume that selectivity can be achieved and increased through targeted interactions within the active site. Furthermore, we hypothesize that the overall sequence differences lead to a specific flexible behavior of PTP1B compared to TC-PTP and that those differences could lead to differing preferred conformations, which can be exploited to design selective inhibitors. Additionally, the concept of dynamically mapping conformational differences to achieve selectivity could be applied to other projects were specificly targeting one of two or more closely related proteins is crucial.

Based on the abovementioned assumptions two approaches are chosen to increase selectivity of PTP1B inhibitors:

I) To detect the key factors of selectivity in and around the active site of PTP1B, crystal structures of the protein in complex with some of the most selective compounds known so far will be investigated thoroughly, starting with

   i) analysis of three-dimensional protein structures with special regard to ligand interactions with sites of amino acid sequence differences to TC-PTP, followed by

   ii) investigations of the flexible behavior of those complexes using molecular dynamics simulations and their comparison to the respective TC-PTP-ligand complexes obtained by homology modeling based on each investigated PTP1B complex structure.

   For the investigations of the protein-ligand complexes established modeling methods like 3D pharmacophores, surface depictions and molecular interaction fields are employed. The investigations of flexible behavior, however, require more elaborate methods. For that, dynamic three-dimensional pharmacophores called *dynophores* [34, 35, 36] as recently established by our group seem suitable. Additionally, due to the high similarity of the two proteins, it seems reasonable to complement this method concentrating on pharmacophoric features by developing a method to assess the quality of steric complementarity of protein and ligand over time. Since this aspect is still neglected in available interaction monitoring tools, such a tool bears the potential for broader application.

II) The second part explores the flexibility of both proteins without a known selective inhibitor with respect to possible differences in conformational preferences: starting with molecular dynamics simulations of both proteins a method will be developed to classify the occuring conformations into clusters of different binding site shapes and identify clusters or conformations preferred by PTP1B, but highly improbable for TC-PTP.

Consequently, the key features of selectivity derived from both approaches will be integrated into an adapted virtual screening workflow to find commercially available compounds with high potential to be PTP1B selective inhibitors.

3. Computational Methods

This work mainly deals with computational methods which can be subsumed under the term Computer Aided Drug Design. Since computational power is developing fast and becoming more affordable, computational methods have become an important part in drug design, saving money and time by limiting the number of compounds submitted to more expensive biological tests [37, 38, 39]. Often different methods are combined to reduce the number of false positive predictions [37]. Depending on the available or utilized data those methods can be divided into ligand- and structure-based methodologies: ligand-based methods are based on 2D structures of known ligands and their activities ranging from quantitative structure-activity relationships, as first developed by Hansch in the 1960s, to three-dimensional pharmacophore generation [40, 41]. Their interpretability is often limited, since predictions are based on two-dimensional ligand structures, elaborated guesses of three-dimensional bound conformations or calculated descriptors based on those structures. Therefore, if available, inclusion of structural information of the protein into the process is preferred.

This chapter briefly describes structure-based methods employed in this study dividing them into the sections ”Structural Data”, ”Conformation Generation” and ”Ligand-Target Complementarity” followed by a section ”Binding Site Shape Clustering” explaining the method to find differences in binding site conformational preferences developed specifically for this project and the underlying principle of conformational selection theory.

3.1. Structural Data

3.1.1. Ligand Databases

Small molecules in commercial databases usually contain information of the molecule together with a two-dimensional representation of the ligand or a string-like encoding of the ligand in SMILES format, where the same chemical moiety can be represented in several ways. Different tools implemented in most modeling software suites as Schrödinger's Maestro [42] or CCG's MOE [43] can be used to convert those structures into a three-dimensional format as required for most modeling tasks. Furthermore, the careful assignment of stereochemistry, tautomeric forms and ionization states is necessary [37, 44].

For preparation of the vendor databases used for screening the Chemaxon Standardizer toolbox [45] was used, since it allows user-defined conversion rules in addition to common preset rules for standardization of the molecular input structures and performs well even on huge databases.

3.1.2. Crystal Structures and Homology Modeling

X-ray scattering and NMR spectroscopy allow the determination of three-dimensional protein structures at almost atomic resolution. However, in common resolution ranges, which are much lower than 1Å, hydrogen atoms cannot be unambiguously assigned based on the experimental data. Therefore, hydrogen atoms are usually assigned based on force-field calculations or common protonation rules. Unfortunately, those methods are not capable to correctly assign unusual protonation states due to surrounding amino acids as the negative charge of the catalytic cysteine in PTP1B, which therefore require manual intervention.

Many of the experimentally derived three-dimensional protein structures are deposited in the publicly available Protein Data Bank [46] together with information about their origin and quality. The selection of crystal structure complexes for PTP1B was driven by the selectivity factor of the complexed ligand and additionally influenced by the resolution of the structures as a simple quality criterion as well as the occurrence of mutations in or close to the binding site that could disturb the investigation of interactions.

If no crystal structure is available, homology modeling can be used to build a three-dimensional structural model from the protein sequence and an experimentally determined three-dimensional structure of a closely related protein. This technique is based on the observation that in protein families, structure is usually better conserved than sequence [47]. The homology modeling process usually starts with the search for a related protein with known 3D structure, followed by the alignment of both sequences, the assignment of the respective coordinates from the template to the target and finally a model refinement step with subsequent evaluation of the model [37]. Homology modeling has been shown to yield useful models for structure based drug design, if the pairwise alignment of target and template exceeds 50% [48]. Common protein structure modeling tools like Modeller [49], PHYRE2 [50] and SWISS-MODEL [51] offer special features like multi-template modelling in addition to automatic template search [52].

For the single template modeling of the known closely related structure of TC-PTP based on PTP1B structures (sequence identity 57%), the modeling tool integrated into the software package MOE [43] was chosen, since it offers detailed control over the modeling process and additionally allows to take into account a ligand bound to the template structure in the modeling process.

3.2. Conformation Generation

3.2.1. Protein-Ligand Docking

In structure based drug design, conformation generation of a ligand often is performed in the form of protein-ligand docking to only obtain ligand conformations that fit the binding pocket of the protein. Available programs address this problem with different methodologies. Differences mainly lie in the way they handle ligand flexibility which can roughly be divided into systematic and random or stochastic approaches [53]. Systematic methods can further be divided into conformational search based, fragmentation based and database based methods [53]. The most thorough, but also computationally most expensive variant of the different systematic approaches is conformational search, where all rotatable ligand bonds are systematically rotated in small steps to evaluate all possible combinations [53]. Fragmentation methods try to circumvent the combinatorial explosion implicated by using conformational search for ligands with a high number of rotatable bonds by docking one or several parts of the ligand and then joining or incrementally growing the solution to a docking pose of the whole ligand [53]. Database based methods seperate the problem of conformation generation from the binding site fitting step, by precalculating a number of conformations per ligand and then rigidly docking those precalculated conformations [53]. The second group of methods to explore ligand conformatinal space employs random or stochastic methods: here Monte Carlo methods and genetic algorithms are the most popular subgroups [53]. Monte Carlo methods select poses based on a probability function, while genetic algorithms start with a population of ligands and by sequentially changing and combining parameters of the population members using a fitness function converge towards a final pose [53, 54]. A prominent program employing a genetic algorithm is the software Gold [54].

The programs also differ in the type of simulation methods they use which can apply principles of molecular dynamics or energy minimization [55]. Possible conformations of the ligand are assessed with scoring functions, which can be theory derived or empirical [56]. Often a simpler function is used for prescoring during the conformation generation process, whereas a more elaborated function is used afterwards to yield more reliable results for affinity prediction [55]. Currently available scoring functions have their limitations, for example are most scoring functions focusing on energetic rather than entropic contributions to binding, which might lead to high ranking errors, if the binding of a concerned ligand is predominantly entropy driven [55]. Therefore, docking programs are able to explore the conformational space of ligands sufficiently well to find binding poses very similar to that found in a crystal structure protein-ligand complex, but in many cases they are not able to correctly place that pose at the top of their ranking [57]. Inclusion of target specific information into the ranking process has been shown to increase prediction quality [57].

A few programs additionally offer the introduction of some degree of conformational flexibility to the target. This can be achieved by either using different input target conformations, known as ensemble docking, or sampling of sidechain conformations, randomly or based on rotamer libraries [55]. However, those artificial changes of a protein need to be handled with care, since introducing protein flexibility to docking can come with the cost of increased false positive rates due to less restrictive binding sites, especially if the cost of the protein movements to these conformations from a low energy conformation is not accounted for [58].

Common docking programs [56] are AutoDock [59, 60], DOCK [61], FlexX [62], Glide [63] and GOLD [54]. For this study, GOLD was chosen, since in a comparative study of docking programs including PTP1B as a target, GOLD performed especially well to recover a high amount of actives with top ranked scores, which seemed not only related to the high negative charges of the known actives for this target [64]. Subsequent energy minimization of the resulting poses together with surrounding amino acids was performed. Careful attention was paid to the fact that energy minimization can introduce hardly recognizable strain in order to optimize directed interactions and therefore create the illusion of good binding [65].

3.2.2. Molecular Dynamics Simulations

A more elaborate way than rotamer changes to sample protein flexibility are molecular dynamics simulations [66]. Those are simulations of the motions of a macromolecule in atomic detail with the aim of assessing its accessible conformational space. This can be done by numerical solution of the classical equations of motion, which is therefore restricted in accuracy by the available computational power as well as the quality of available force-fields and includes the following components [67, 68]:

> **I: System Set-Up.** The input structure is inspected and errors are corrected, the ionization state of the macromolecule is calculated, counter ions and solvent are included.
>
> **II: Force Fields.** Forces are calculated for every atom with the help of force-field equations. Those equations contain parameters for bond stretching, bending and rotations as well as for non-bonded interatomic interactions like van der Waals contacts and electrostatic potentials.
>
> **III: Laws of Motion** Using the previously calculated forces accelerations and velocities are computed using Newton's law of motion. Initial velocities are usually assigned randomly based on the overall energy of the system and therefore replicas of the simulation using different initial velocities help to sample the effects of different starting points.
>
> **IV: Trajectory Simulation.** With the obtained velocities atom coordinates can be updated. However, since the calculations include numerical integration, this needs to be done for a time step of shorter than the fastest movements in the molecule and is usually preset between 1 and 2 fs. With the repetition of steps II to IV snapshots of atom coordinates can be saved over a period of time to form the trajectory.

Common software for molecular dynamics simulations includes AMBER [69], GROMACS [70], Desmond [71] and NAMD [72] and OpenMM [73]. Simulations by those programs are to a great extent depending on the applied force-field and the model chosen to represent the solvating water. However, no force-field could be shown to be consistently more feasible for drug design approaches than the others and simulations on one starting structure with different force-fields often show consistent results [74]. For this study the freely available Desmond software with OPLS-AA force-field was chosen due to its user-friendliness being integrated into the Maestro software suite and its ability to assign ligand parameters [71].

Initial analysis steps usually involve calculation of root mean square deviation (RMSD) and root mean square fluctuation (RMSF). The RMSD is calculated as follows

$$RMSD = \sqrt{\frac{1}{n}\sum_{i=1}^{n} d_i^2}$$

with n: number of atoms compared and d_i: distance of one atom i in one frame compared to a reference frame.

RMDS calculation requires a preceding alignment step of each frame to a reference structure - usually the starting structure of the simulation - in order to eliminate the effects of translational and rotational movements of the whole protein during simulation. Plotting of the RMSD of the C_α atoms of a protein over the time of a molecular dynamics simulation allows conclusions about the stability of a protein structure.

The RMSF describes the atom-wise deviation to a reference structure - usually a mean structural state of the protein over the simulation or the starting structure of the simulation - averaged over the simulation steps. This allows to identify and compare stable and unstable regions of a protein over a simulation.

3.3. Ligand-Target Complementarity

Affinity of a ligand to a target is heavily affected by its favorable and repelling interactions. Therefore, investigating these interactions can lead to important insights for the design of new ligands. The main principles found in almost all high-affinity protein ligand complexes are high steric complementarity, high complementarity of surface properties like polarity or hydrophobicity and an energetically favorable ligand conformation [75]. However, some of those criteria are easier to assess then others. For example the entropy contribution to binding free energy is usually not observable in static structures [76]. Furthermore, a protein-ligand pair can display more than one stable bound conformation [76]. Additionally, flexible protein parts tend to prefer more flexible ligand moieties, another aspect hard to observe in static structures [65].

Several concepts have been applied to visualize favorable protein-ligand interactions or promising interaction sites at a protein surface. The ones that play an in important role in this study, namely (1) molecular interaction fields, (2) 3D pharmacophores and dynophores as well as (3) shape complementarity calculations will be described below.

3.3.1. Molecular Interaction Fields

Molecular interaction fields (MIFs) depict the spatial distribution of the interaction potential between a target structure and a probe [77, 44]. That means for a given probe, like a secondary amine nitrogen, this probe is placed at every point of a regular grid in and around the target structure where its energy of interactions is calculated taking into account for example the hydrogen bonding energy, van der Waals forces as well as charge interaction energies. This data can then be depicted in form of an interaction map at a chosen interaction energy threshold for a specified probe. Which terms of interaction energy and which parameters of the probe are included into the calculations depends on the implementation of this method, of which the most elaborate one is found in the software GRID [78, 79]. Since this method was only used to support the screening model developed based on the binding site shape clustering explained below, the readily available implementation of the MOE software was considered of sufficient accuracy.

The application of molecular interaction fields was selected to support screening in this work, since the developed shape pattern alone would not have been restrictive enough for virtual screening. Additionally, the shape pattern was designed to introduce selectivity, which cannot be achieved without activity. Consequently, it was aimed to introduce activity based on the information on favorable protein-probe interactions. To combine both the selectivity and the activity model, the format of a 3D pharmacophore (described in the following section) was chosen, since 3D pharmacophores are particularly efficient for virtual screening [80, 81]. This, however, requires the translation of molecular interaction field potentials into pharmacophoric features, which was achieved by calculating the local minima of the computed interaction fields for different probes reflecting hydrogen bond donor, hydrogen bond acceptor and hydrophobic behavior.

3.3.2. 3D Pharmacophores and Dynophores

3D pharmacophores are abstractions of interactions into different types like hydrogen bonding interactions, hydrophobic contacts and aromatic interactions together with their spatial arrangement. In a stricter sense like specified in the IUPAC definition, the term 3D pharmacophore only describes those kinds of three-dimensional interaction patterns for which their containing features are ”necessary to ensure the optimal supramolecular interactions with a specific biological target structure and to trigger (or to block) its biological response” [82]. In order to ensure that an interaction pattern corresponds to that definition, validation of the model is performed by assuring a good compromise of retrieval rates of known active and potentially inactive molecules [75, 83]. Nevertheless, a pharmacophoric feature found with all known ligands does not mean it is necessary to achieve high affinity binding as well as good pharmacophoric fit can still lead to low affinity due to e.g. steric clashes [75]. Due to their high level of abstraction, pharmacophores are especially feasible for scaffold hopping [84].

Common programs for 3D pharmacophore generation and virtual screening are CATALYST [85], Phase [86], MOE [43], LIGANDSCOUT [80] and FLAP [87]. They show subtle distinctions in feature definition and placement, as well as more significant differences in the matching algorithm used for screening, which affects accuracy of the results as well as screening velocity [88, 84]. In this study the MIF-based interaction patterns were encoded into LigandScout pharmacophore format, since it is easily interpretable and manipulatable and allows fast and accurate screening.

Recently, the concept of 3D pharmacophores derived from a protein structure and a bound ligand was transferred to molecular dynamics simulations of protein-ligand complexes [34, 35, 36]. This allows the analysis of protein-ligand binding for many different conformations and therefore a more detailed inspection of the stability and quality of interactions together with the variability of interaction partners on the protein site. For this purpose, the Dynophore application was used for analysis of molecular dynamics simulations of higher selective ligands in PTP1B in comparison to simulations of the same ligands in homology models of TC-PTP.

3.3.3. Shape Complementarity

The abovementioned tools for analyzing ligand-target complementarity only cover one of the three principles of high affinity binding, the high complementarity of surface properties. Unfortunately, steric complementarity is not sufficiently accounted for in established methods for monitoring protein-ligand interactions. 3D pharmacophores, however, often allow specification of excluded volumes, which enable the user to add spacial restrictions to the interaction pattern and can introduce steric complementarity of the hits to the target [84]. But especially for shape complementarity, assessment on a single snapshot seems of limited use, due to for example the previously mentioned tendency of flexible protein parts to prefer more flexible ligand moieties. Therefore a tool was created and implemented in R that can quantify shape complementarity throughout a whole molecular dynamics trajectory, trace back the complementarity to certain parts of the ligand and allows the statistical analysis of differences for one ligand in different protein surroundings [89]. Nevertheless, a particularly good shape fit can be less energetically favorable, if the ligand is forced into this conformation due to the lack of better alternatives. In order to detect especially unfavorable ligand conformations, the ligand strain energy was calculated in addition to the shape fit and they were assessed together. Strain energy is an estimate of the difference between the energy of the ligand in the bound state compared to that in the nearest local energy minimum conformation. The strain energy can be calculated as a MOE 3D descriptor. However, due to the many simplifications and assumptions integrated in this calculation, the generated values need to be handled with caution. They are therefore not accurate enough to be integrated in calculations of ligand binding energy, but still can give important hints on the twist or strain of a ligand.

3.4. Binding Site Shape Clustering

The method of binding site shape clustering developed in this study to exploit differences in conformational flexibility of two closely related proteins is based on the concept of conformational selection theory. It is assumed that conformational changes in the target happen before association with a ligand and that the ligand chooses an appropriate conformation for binding out of the available ensemble of target conformations. Protein-ligand binding therefore gets likelier with increased presence of suitable protein conformations in the ensemble of target conformations as well as with increased presence of suitable ligand conformations in the ligand conformational ensemble. This is in line with the abovementioned increased probability to find energetically favorable ligand conformations in high-affinity protein-ligand complexes. Conformational selection theory therefore stands in contrast to the concept of induced fit assuming initial ligand binding to a suboptimal protein conformation followed by an adaption of the protein to the bound ligand. Recent studies describe increasing evidence for the existence of ligand binding conformations without ligand presence and assume a dominant role of conformational selection, although the coexistence of both mechanism cannot be ruled out [90, 91].

The screening process implemented in this study additionally considers ligand rigidity as favorable. Chang et al. state that a ligand binding to a protein loses conformational flexibility resulting in an entropy loss opposing binding [92]. Some sources therefore assume that binding of an inflexible ligand is less entropically unfavorable, since rotational freedom of the ligand was already restricted before binding [93]. However, experimental evidence for this assumption is scarce, which may be due to the lack appropriate methods for determination of entropic contributions to binding in such detail. Chang et al. additionally stress the downside of rigid ligands: they require an exact fit of the protein to the ligand and therefore high resolution structures and accurate predictions in the modelling process, since the ligand's ability to adapt to the binding site is limited [92].

Based on those assumptions, the method of binding site shape clustering was developed: To compare protein conformations with focus on the catalytic binding site surroundings, the open source tool POVME2 [94, 95], originally created to measure and compare pocket volumes, is used to depict the shape of the binding site for every molecular dynamics frame based on equidistant points. Since the tool is used to compare the shapes of the resulting point maps, a preceding alignment step of the molecular dynamics frames is required. This alignment influences the shape point maps, since the placement of the initial map points in space stays the same for every frame, only those points of a bigger map that are very close to or inside the protein are deleted.

The resulting shape point maps are then translated into bit strings encoding presence or absence of each point of the starting point map in the binding site shape point map for each molecular dynamics frame respectively using the software R. If molecular dynamics trajectories of two similar proteins are aligned and binding site shape point maps are calculated with the same starting point map, the bit strings of both proteins can be compared. To identify the most prominent conformational states of the binding sites throughout the molecular dynamics simulations, cluster analysis of the bit string data was performed. Clustering methods have become a popular means to deal with the high amount of data from molecular dynamics simulations in different contexts [96, 97, 98].

Cluster analysis seeks to create homogeneous groups of objects with low inter-cluster homogeneity in a set of data based on the states or values of their attributes. Clustering therefore is a valuable tool to structure data, although it has to be considered that the resulting clusters do not necessarily have a useful interpretation for the analyzed aspects of the data. Additionally, very different groups can be found for the same set of data using different clustering algorithms [99]. Since the final cluster size or number of clusters to obtain was unclear, a hierarchical clustering algorithm was chosen [99]. The term hierarchical clustering covers methods that divide data into a hierarchy of groups with levels of subgroups and therefore allows for the decision of cluster size to be postponed until after clustering. Hierarchical clustering methods can further be divided into agglomerative methods, starting with singletons and merging clusters until all data is combined in one cluster, and divisive methods, starting with the whole dataset and subsequently dividing this set into smaller groups [99]. The chosen clustering method falls within the category of hierarchical agglomerative clustering methods, because it starts with each entry being its own cluster and consequently merges clusters in a way that the increase in dissimilarity sum stays minimal [100].

As clustering algorithm, the function ward.D from the package hclust was used as implemented in R which is a generalization of Ward's clustering algorithm [100]. The Ward algorithm was developed for data that allows calculation of Euclidean distances and then depicts minimization of the variance of the data in a cluster. Finch showed that calculating appropriate distance measures for dichotomous data and submitting the calculated distance matrices to Ward's clustering can lead to good results representing the natural groups of the data. However, it has to be considered that the original interpretation of minimizing variance is lost, if other than Euclidean distances are used as input [101]. For calculating the similarity/dissimilarity of two objects the simple matching coefficient was used here, which is defined as the number of matching attributes divided by the number of total attributes [102]. This differs for example from the Russel-Rao Index where only the simultaneous presence, of a state is considered as matching, but not the simultaneous absence. It was reasoned here that absence of a point in the map should be weighted the same as its presence, since the total map size stays the same during the calculations and considering both simultaneous appearances as matching is therefore not resulting in different weighting of "off" or background attributes for different frames as had to be considered if the starting map would have been small in some cases and big in others. Nonetheless, the absence of a point which is never "on" and therefore never considered part of the binding site is rated as a similarity with this measure. This was considered negligible since this number should be small due to the restricted binding site definition.

After clustering, the cluster stage (total number of clusters) useful for analysis is chosen with a variation of the elbow method [103]. The clusters of the selected stage are further processed: In order to differentiate between conformations possible for both proteins and those only possible for one of them, an affiliation ratio representing the number of frames from PTP1B divided by the total number of frames in that cluster was assigned to each of them. Additionally, occupancies are calculated for each point of the starting point map throughout all frames, representing how often a specific point is marked as present (1) in the binding site divided by the total number of frames in that cluster.

Differences of the clusters are then computed to find the PTP1B affiliated cluster that is most different to all TC-PTP affiliated clusters. After selection of this cluster potentially representing a PTP1B exclusive conformation, the points with significant differences in occupancy in this cluster compared to all TC-PTP clusters are extracted and analyzed. They are further used to extract a diverse selection of frames matching this selectivity map, which can then be used for virtual screening as described in the previous sections.

All steps processing the POVME2 output were collected in R scripts to ensure easy adaption and repeating of the steps on different samples.

4. Results

4.1. Part I: Protein-Ligand Interaction Analysis

4.1.1. Sequence Comparison of PTP1B and TC-PTP

Sequence alignment with Clustal Omega [104] calculates a total sequence identity of about 57%. The binding sites of both protein tyrosine phosphatases, however, are highly conserved: 55 residues were considered as binding site residues and compared. According to Li et al. they were grouped into different subsites named site A (catalytic site) to site D [105]. The catalytic site shows a sequence identity between both proteins of over 95%, while all subsites together still show a sequence identity of 80%. Table 4.1 highlights differences in amino acid sequence for the binding site subsites of PTP1B compared to TC-PTP.

Subsite	Residues	Sequence PTP1B	Sequence TC-PTP
Site A	T177-P185	TTWPDFGVP	TTWPDFGVP
	H214-R221	HCSAGIGR	HCSAGIGR
	T263-Q266	TADQ	TPDQ
Site B	R254-Q262	RKFRMGLIQ	RKYRMGLIQ
	R24-C32	RHEASDFPC	RNESHDYPH
	Y20	Y	Y
	F52	F	Y
Site C	R47-P51	YRDVSP	YRDVSP
	K41	K	E
Site D	K116-C121	KGSLKC	KESVKC
	R45-Y46	RY	RY

Table 4.1.: Sequence comparison of active site surrounding residues (PTP1B numbering). Differences are highlighted in red.

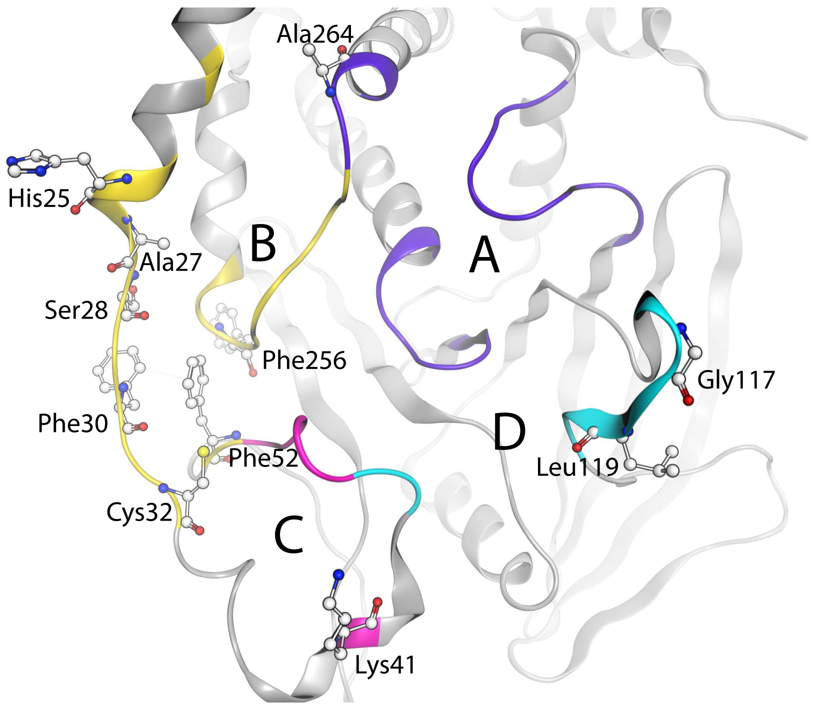


Figure 4.1.: PTP1B (from crystal structure 1PTY) with sequence differences to TC-PTP and binding site subsites.

As becomes obvious from the sequence comparison, the subsites surrounding the catalytic cavity, named B, C and D, contain several differences in sequence in PTP1B compared to TC-PTP. To assess their relevance for ligand binding, they were also highlighted in a three-dimensional depiction of the PTP1B protein structure (see Figure 4.1).

Unfortunately, most of those differences seem of minor relevance for the design of selective inhibitors: A264 and the respective proline found in TC-PTP are turned away from the cavity and could therefore only influence the space available for the ligand in the binding site. Due to their similar size, however, this difference is insignificant. F256 and its corresponding tyrosine sidechain are similar in size and faced inside the protein, therefore the additional hydroxyl group in TC-PTP cannot be involved in ligand binding. Due to the orientation away from the pocket towards the solvent an exchange to a valine for Leu119 also seems unlikely to influence the binding cavity, the same holds for Gly117 and its corresponding glutamic acid residue. However, due to their differences in size and polarity, changes in the flexible behavior of the D site loop in TC-PTP can be expected. The remaining differences are mainly concentrated in the B site of the binding area and although some residues like histidine 25 are faced away from the binding pocket, the high amount of differences in this area is likely to result in at least subtle differences of the pocket's shape and electrostatic properties. Furthermore, Lys41 in the C site could lead to a different conformation of the adjacent C site (YRD) loop through charge interactions not possible for the corresponding valine in TC-PTP.

4.1.2. PTP1B Crystal Structure Analysis

With the aim of getting insight into structural features for selectivity of PTP1B ligands against TC-PTP, crystal structures of PTP1B were analyzed with emphasis on structures with co-crystallized ligands known to show some selectivity towards PTP1B.

Selectivity	*PDB Structure*	*Activity Type*	*Activity in PTP1B in nM*	*Activity in TC-PTP in nM*	*Selectivity Factor*	*Source*
selective	2F70	IC_{50}	33500	203500	6.1	[106]
	2F6T	IC_{50}	42500	169000	4.0	[106]
	1XBO	IC_{50}	920	19200	20.9	[107]
	1QXK	K_i	9000	182000	20.2	[107]
	1Q6T	IC_{50}	5	36	7.2	[107]
	1Q1M	K_i	6900	164000	23.8	[107]
	1PYN	K_i	3200	24780	7.7	[107]
	1PH0	K_i	120	470	3.9	[107]
	1NZ7	K_i	76	380	5.0	[108]
	1NNY	K_i	22	49	2.2	[107]
unselective	2B07	K_i	370	380	1.0	[109]
	1Q6P	IC_{50}	3	3	1.0	[107]
	1NL9	K_i	1100	1100	1.0	[107]
	1ECV	K_i	14000	14000	1.0	[107]

Table 4.2.: PTP1B crystal structures selected for analysis with ligand activities in PTP1B and TC-PTP and the resulting selectivity factor.

Systematic investigation of PTP1B crystal structures was started with 145 crystal structures of human PTP1B listed in the Uniprot database for human PTP1B (P18031) at the time of this study (last checked March 2018), 7 of which were excluded from analysis due to low resolution (>2.70 Å) [110, 46]. Additional 59 structures were excluded due to covalent modifications in or close to the active site. From the remaining 79 crystal structure complexes 14 complexes were chosen, for which activity data of the ligands towards both PTP1B and TC-PTP were available and where the ligands showed either selective behavior against TC-PTP (selectivity factor > 2) - true for 10 of the structures - or unselective behavior (selectivity factor between 0.95 and 1.05). Table 4.2 lists the selected structures with their ligand activities in PTP1B and TC-PTP and the resulting activity factor. Interaction counts for polar interactions, counting hydrogen bonds as well as ionic interactions as 1 and water mediated hydrogen bonds as 0.5, are depicted as heat map in Figure 4.2(left). On the one hand, unselective ligands show a tendency for fewer B site interactions to Gln262, Arg24 and Arg254. On the other hand, polar interactions to the sidechain of Asp48 and Asp181 seem favorable to introduce selectivity. In the investigated complexes selective ligands additionally show a tendency towards fewer interactions to the backbone of Phe182. For complexes 1Q6T and 1Q6P crystal packing seems to influence the interactions of the ligand to the protein. Conclusions from interaction counts for those structures therefore show a higher level of uncertainty.

The counts of hydrophobic parts of the ligand interacting with hydrophobic areas on the protein surface depicted in Figure 4.2(right) show no clear tendency of the interactions to increase or decrease selectivity of the ligands. However, during the investigation of the crystal structures the missing ability of this method to distinguish between bigger and smaller lipophilic areas of interaction was noticed.

4.1.3. TC-PTP Crystal Structure Analysis and Homology Modeling

Only one TC-PTP crystal structure was available in the PDB at the time of this study (PDB code: 1L8K, resolution 2.56 Å; last checked March/2018). Figure 4.3 shows this TC-PTP structure superposed to the PTP1B structure 1PTY depicting only the backbones of the two protein structures. The main differences concerning the ligand binding site are found in the conformation of the catalytic cavity loop depicted in violet. This loop is found in an open position in the TC-PTP structure. This open conformation is also possible for PTP1B, but the loop closes for both proteins upon substrate binding. Additionally, he D site loop shows significant conformational differences with a

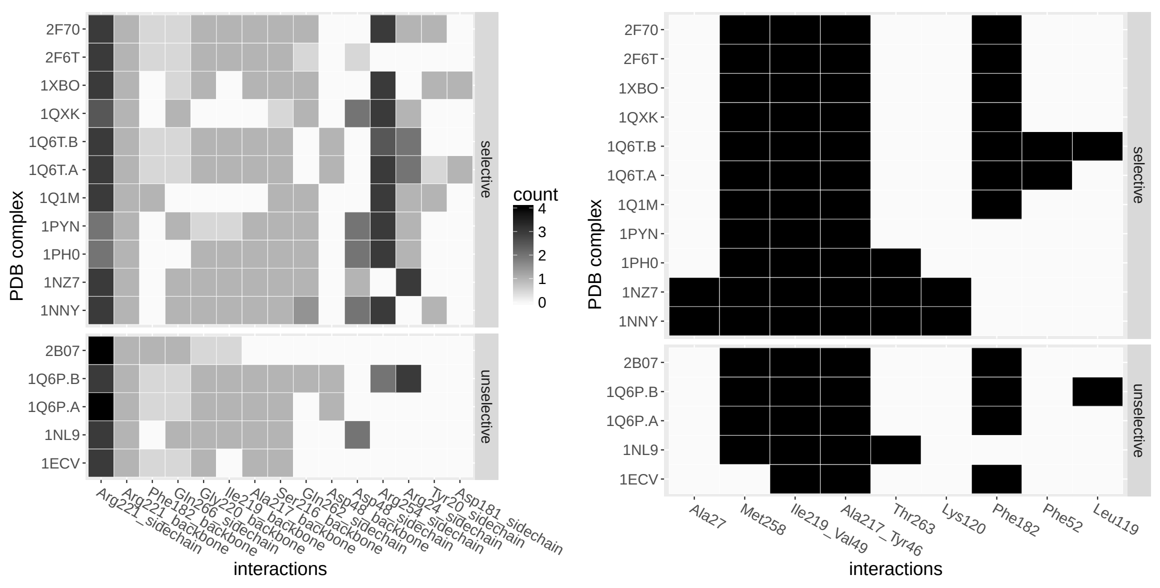


Figure 4.2.: Interaction plots for protein-ligand binding in selected PTP1B complexes. Left: polar interactions shaded by interaction count, right: hydrophobic interactions (black = presence of interaction).

β-sheet structure in PTP1B, but not in TC-PTP. However, comparison of several PTP1B structures revealed conformational variations for this loop, which shows disordered behavior in the original publication of the TC-PTP crystal structure [20]. The publication further suggests an unusual conformation introduced by crystal packing for this loop.

Since this assessment led to the conclusion that the available TC-PTP crystal structure was not feasible for detailed structure comparisons regarding selective PTP1B ligands, homology models were created for TC-PTP-ligand complexes based on each of the three most selective PTP1B crystal structures 1QXK, 1Q1M and 1XBO. From the ensemble of structures produced during homology modeling, conformations were chosen that show high resemblance to the respective PTP1B structures regarding sidechain and ligand conformations. Figure 4.4 shows one of the TC-PTP homology models in comparison to the PTP1B template.

4.1.4. Molecular Dynamics Simulations

Homology modeling for TC-PTP resulted in three-dimensional structures very similar to the PTP1B templates. In order to elucidate whether the complexes show differences in their flexible behavior or if even equilibration of the TC-PTP models would lead away

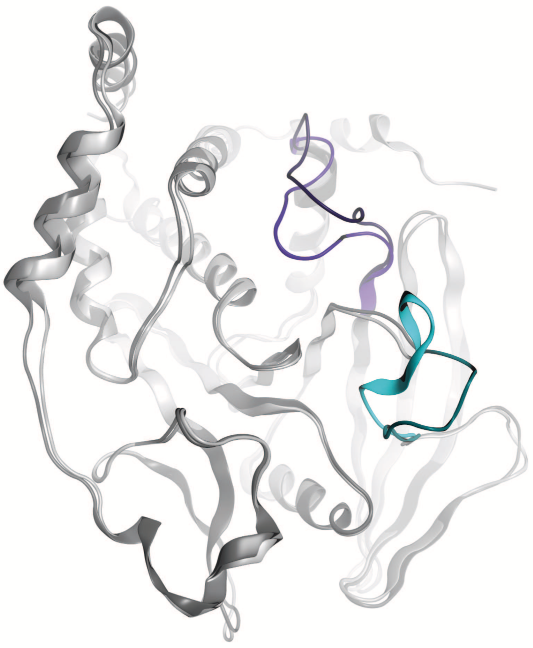


Figure 4.3.: PTP1B (PDB code 1PTY) in light grey superposed to TC-PTP (PDB code 1L8K) in dark grey. Major conformational differences highlighted in violet and cyan.

from the PTP1B equivalent ligand binding conformation, molecular dynamics simulations were performed and analyzed. For each of the three most selective PTP1B complexes and the respective homology models three simulations with varying seeds were performed to be able to distinguish effects due to the initially assigned velocities from system typical behavior during analysis.

For the 1XBO complex and the respective TC-PTP homology model complex, analysis of the C_α-RMSD plots after backbone alignment (Figure 4.5) showed stable system behavior with similarly fast equilibration of the PTP1B and TC-PTP systems after less than 5 ns and only minor deviations up to 2.1 Å from the starting conformation. Simulation 1 of the homology model shows a slight RMSD drift towards the end of the simulation time. Overall, the RMSD plots show a tendency of the homology models for slightly higher RMSD values, which is in good agreement with the expectations considering that TC-PTP was forced into a PTP1B conformation during homology modeling. Similar behavior is found for the 1Q1M complex and the respective TC-PTP homology model complex (Figure 4.6). However, the differences between PTP1B and TC-PTP are smaller, suggesting that the homology model based on 1Q1M is slightly closer to a native TC-PTP structure.

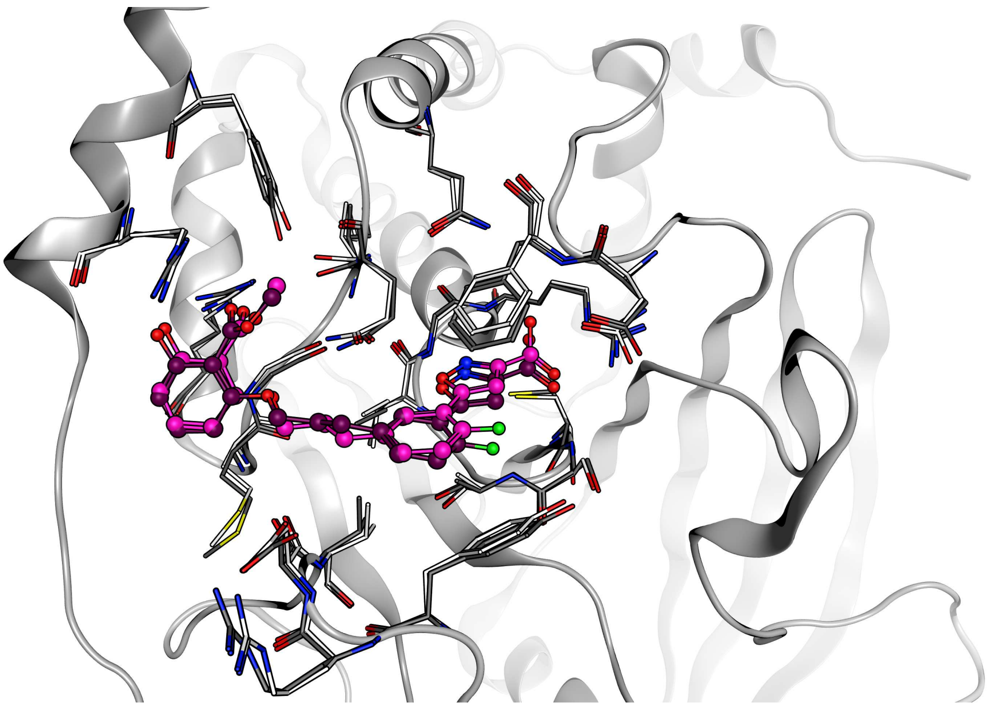


Figure 4.4.: Homology model of TC-PTP (light) based on the PTP1B complex 1Q1M (dark).

The simulations of the 1QXK complex and the respective TC-PTP homology model complex show slightly increased RMSD values compared to the simulations of the other complexes with maximum RMSD values below 2.0 Å (Figure 4.7). The plot shows that all structures finally move away from an initial RMSD plateau after at maximum 10 ns. Apart from simulation 2, the homology model simulations of 1QXK show more stable behavior than the respective PTP1B simulations, although with higher initial RMSD values for the homology model. Simulation 2, however, seems to finally join the equilibrium of the other replica after about 15 ns, despite the increased initial RMSD values.

Additionally, RMSF values were calculated and plotted (Figures 4.8 to 4.13) to determine flexible and inflexible regions and compare those areas for both proteins: All simulations show a high correlation of stable and unstable regions between the different complexes, but also between PTP1B and TC-PTP simulations. This distribution of flexible and inflexible regions is in good agreement with the secondary structure elements of the proteins. Unsurprisingly, the C and N terminal regions show slight increase of flexibility. For the 1QXK simulations (Figures 4.12 and 4.13) the C terminal end shows highly increased flexibility with RMSF values greater up to more than 7 Å. The biggest most stable region for all simulations is the catalytic loop containing the catalytic cys-

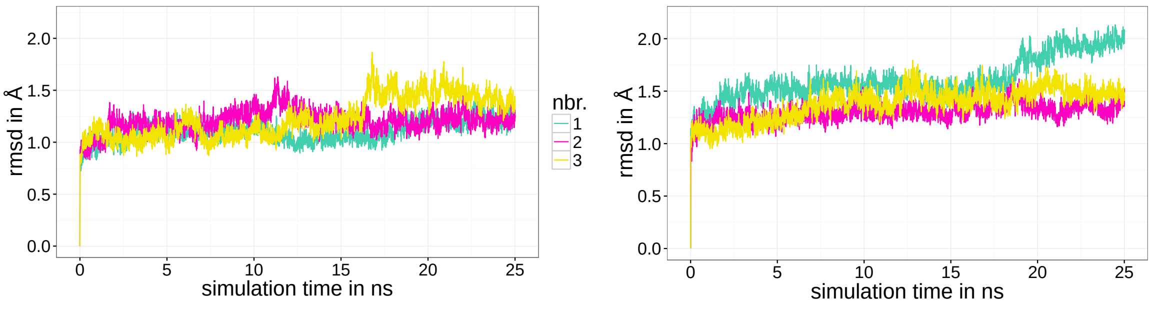


Figure 4.5.: RMSD plots of the molecular dynamics simulations performed on the 1XBO crystal structure complex (left) and the respective TC-PTP homology model complex (right).

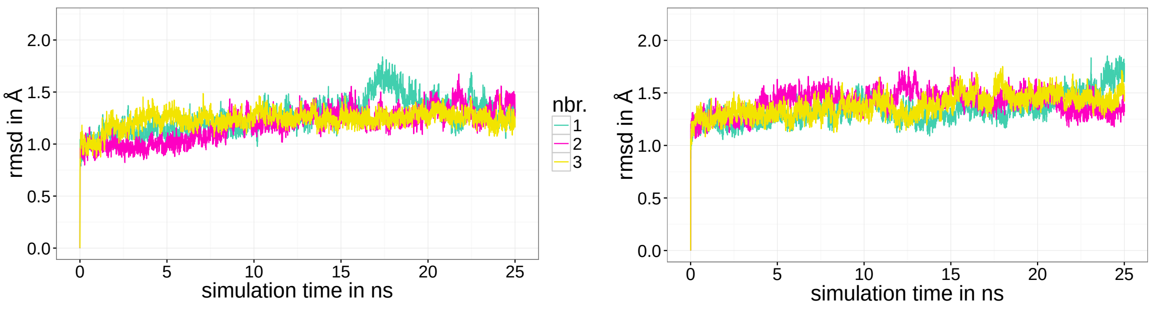


Figure 4.6.: RMSD plots of the molecular dynamics simulations performed on the 1Q1M crystal structure complex (left) and the respective TC-PTP homology model complex (right).

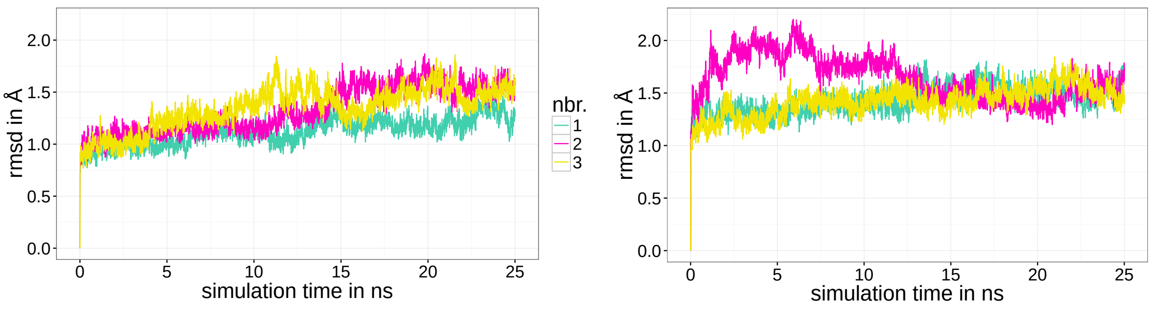


Figure 4.7.: RMSD plots of the molecular dynamics simulations performed on the 1QXK crystal structure complex (left) and the respective TC-PTP homology model complex (right).

teine and the α helix connected to this loop. Interestingly, the WPD loop, known for its high flexibility during the ligand binding process, shows only limited flexibility below 2 Å for all simulations. The B site residues from 254 to 262 as well as the YRD loop (47-51) show slightly increased flexibility compared to the catalytic loop for all simulations. Additional to those similarities, some cases of behavior specific for PTP1B or TC-PTP can be observed from the RMSF plots: The 1XBO and 1QXK simulations are showing higher flexibility for the TC-PTP simulations regarding the D site (116-121) and the neighboring region (122-140). Additionally, for all complexes, the TC-PTP simulations show higher flexibility in the residues 30-40, a loop like structure lying directly behind the YRD loop viewed from the catalytic cavity.

4.1.5. Dynophore Analysis

In addition to the abovementioned parameters, dynamic pharmacophores were calculated and analyzed to gain insight into protein-ligand interactions to further characterize the selective complexes and their stability over time. For this analysis, frame 1000 was considered as starting point of the dynophore analysis, since all systems were considered equilibrated based on the RMSD analysis after this point (about 5 ns).

Figure 4.14 shows the three-dimensional depiction of the first simulation based on the 1XBO complex. The three-dimensional depiction of a dynophore is influenced by the alignment of the single frames, in this case the C_α-atoms of the stable active site loop amino acids Cys215-Arg221 were chosen. Since no differences are obvious between PTP1B and TC-PTP on visual inspection of the dynophores, the underlying interactions were statistically analyzed: To elucidate the differences between PTP1B and TC-PTP ligand binding features, two-dimensional plots of the ligands together with the detected hydrogen bond acceptors (HBA), hydrogen bond donors (HBD), hydrophobic (HYD), aromatic (AR) and negative ionizable (NI) features were created. For each feature mean occurrences over three molecular dynamics simulations together with the 95% confidence intervals were plotted as bar charts juxtaposing PTP1B and TC-PTP.

For all three selective complexes, differences between PTP1B and TC-PTP are small compared to the confidence intervals. However, for all simulations, mean occurence values of the B site hydrogen bond acceptors (HBA5, HBA6 and HBA7) tend to be higher in PTP1B simulations compared to the mean values of the corresponding TC-PTP simulations. This tendency is even more distinctive for the hydrophobic B site interaction (HYD1).

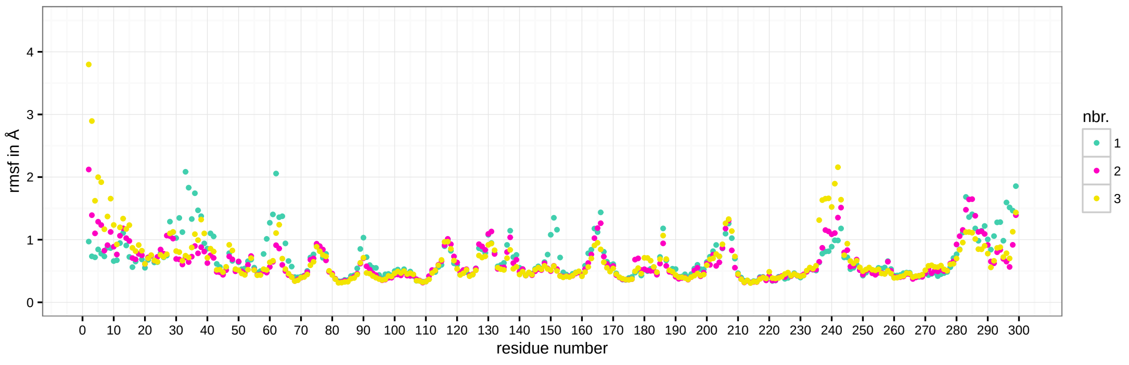


Figure 4.8.: RMSF plot of the molecular dynamics simulations performed on the 1XBO crystal structure complex.

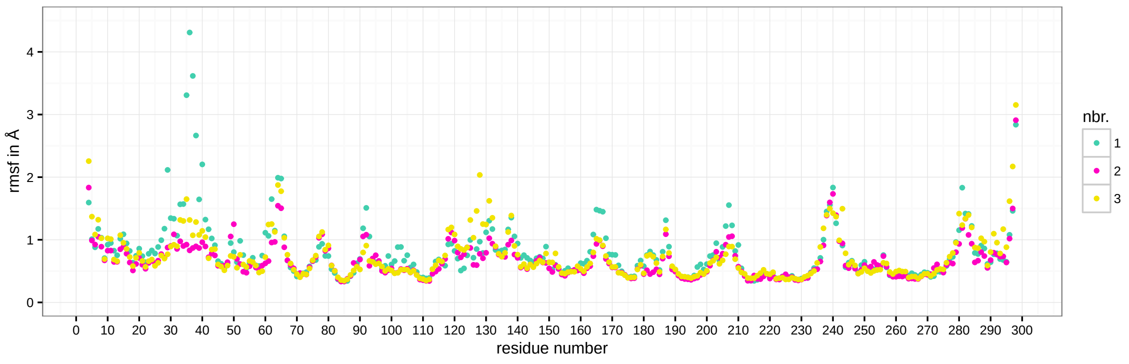


Figure 4.9.: RMSF plot of the molecular dynamics simulations performed on the TC-PTP homology model complex based on 1XBO.

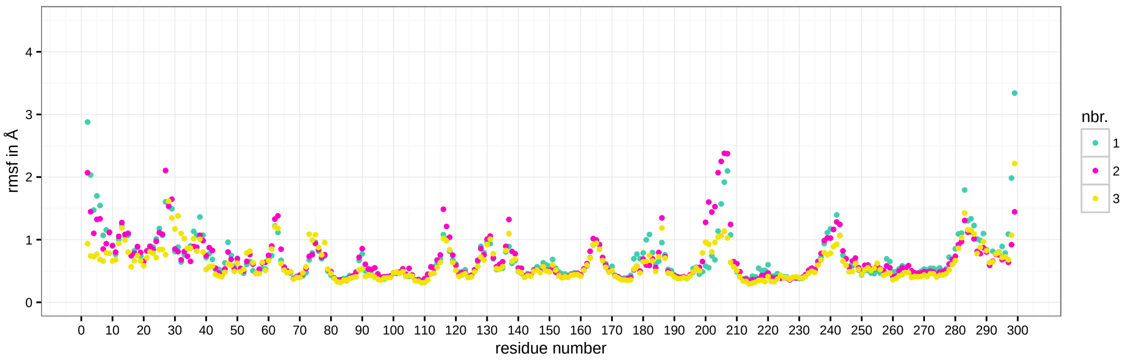


Figure 4.10.: RMSF plot of the molecular dynamics simulations performed on the 1Q1M crystal structure complex.

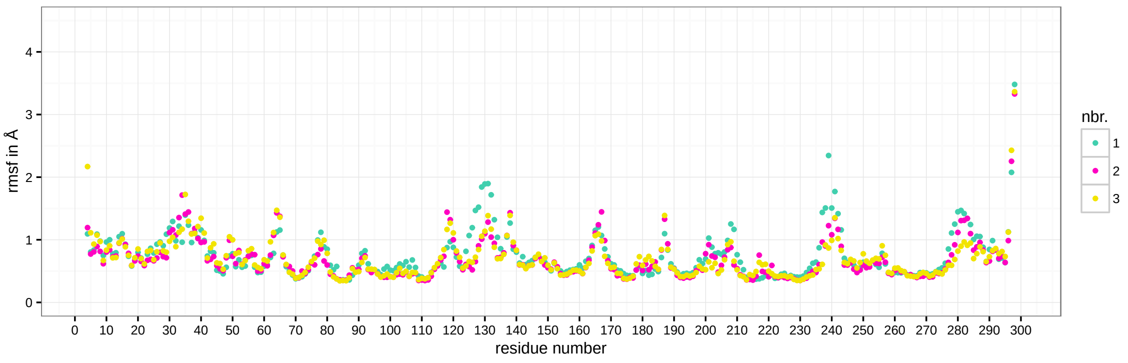


Figure 4.11.: RMSF plot of the molecular dynamics simulations performed on the TC-PTP homology model complex based on 1Q1M.

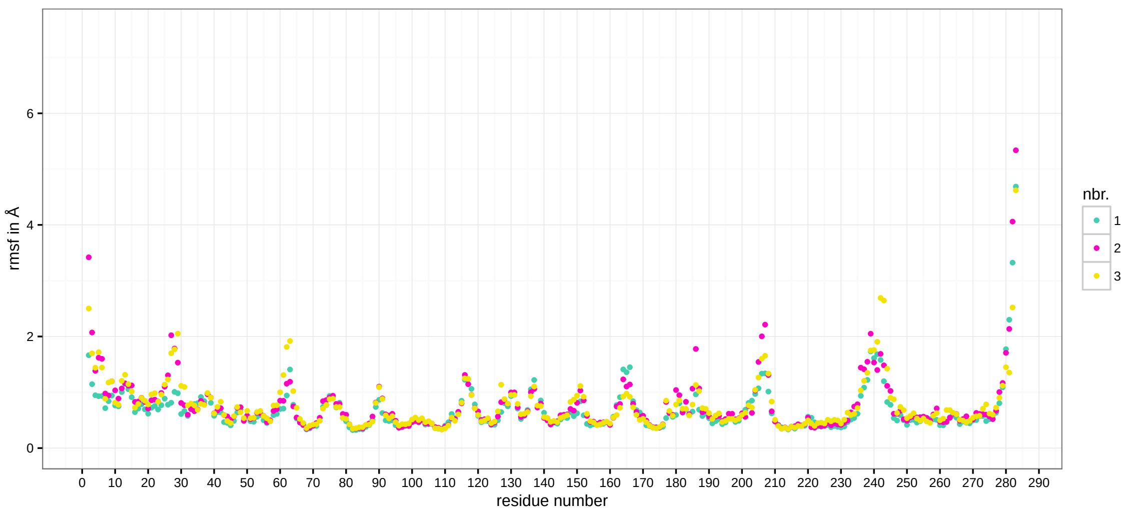


Figure 4.12.: RMSF plot of the molecular dynamics simulations performed on the 1QXK crystal structure complex.

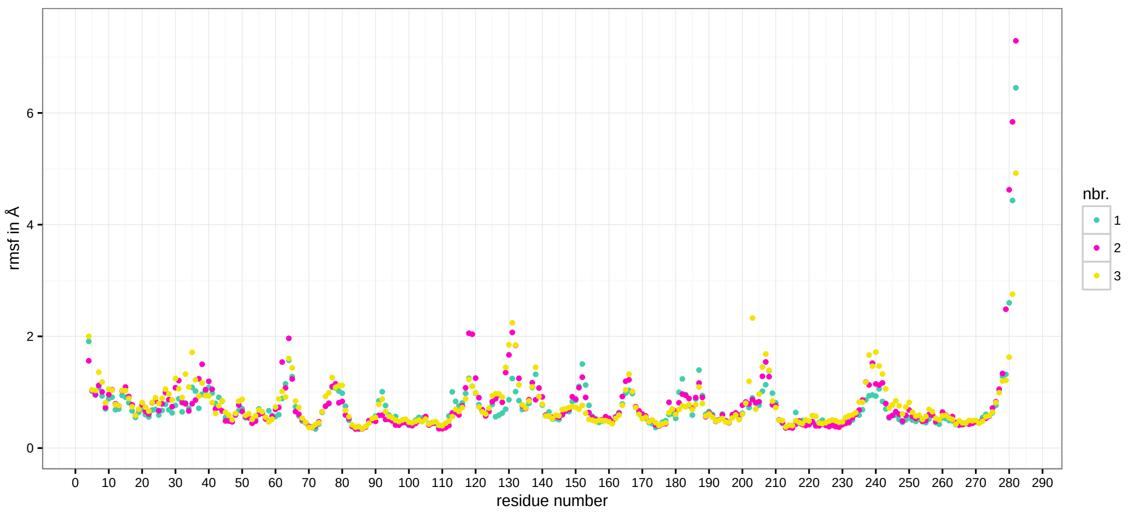


Figure 4.13.: RMSF plot of the molecular dynamics simulations performed on the TC-PTP homology model complex based on 1QXK.

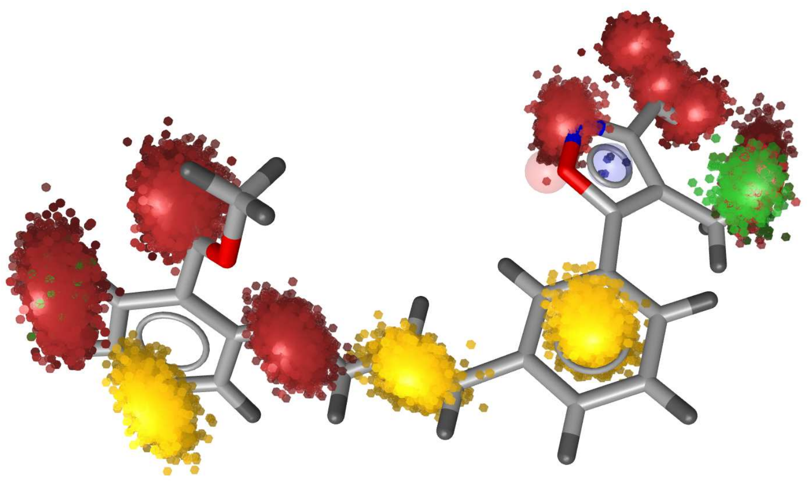
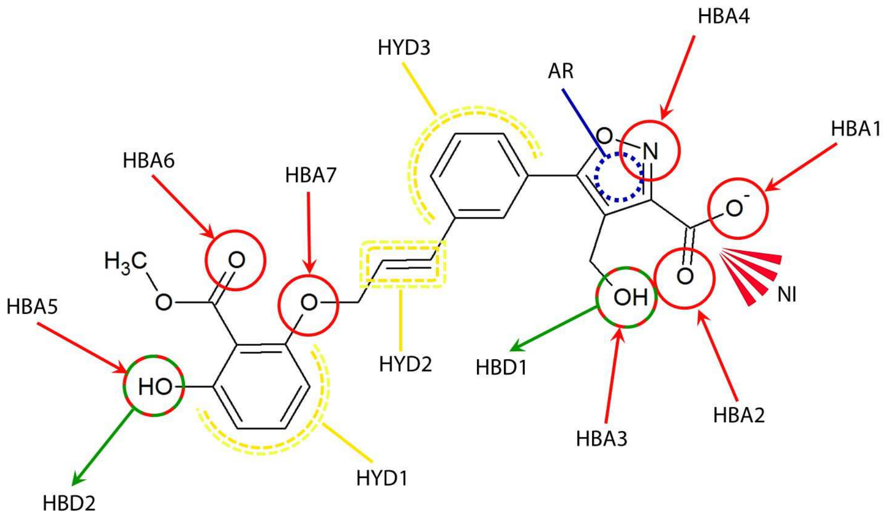


Figure 4.14.: Three-dimenional depiction of the dynophore created based on one of the performed molecular dynamics simulations of the 1XBO complex. Dots represent pharmacophoric features of interactions detected for a molecular dynamics frame. Red: hydrogen bond acceptor or negative ionizable interaction, green: hydrogen bond donor, blue: aromatic interaction, yellow: hydrophobic interaction.

Interestingly, there is one interaction in the catalytic cavity - HBA3 - which shows comparably high occurrence in PTP1B compared to TC-PTP for the simulations of 1XBO and 1Q1M.

Apart from those similarities, the dynophore interactions also reveal binding interactions characteristic for each single protein: The simulations based on the 1XBO complex show the fewest, but most stable interactions. The dominant interaction partner in the only occasionally occurring HBD2 interaction are different water molecules suggesting minor importance for the protein ligand interaction energy. Closer analysis of the interaction partners on protein side for all interactions reveals a tendency of the PTP1B interaction HYD1 to interact not only with Met258, but also Ile219, whereas the latter interaction is not found for the TC-PTP simulations. Instead, in the TC-PTP simulations the Met interaction is extended to the HYD2 feature. The high occurrence of the HBA4 interaction was found to be caused by a trapped water molecule transferred from the crystal structure. The 1Q1M simulations show more interactions, some of them, especially the aromatic interactions, are of very low occurrence. As for the 1XBO simulations, the A site interactions (NI, HBA1 and HBA2) as well as the hydrophobic features are of high occurrence, suggesting very stable hydrogen bonds with active site amino acids and good placement of the ligand in the channel, connecting A and B site of the binding pocket with increased flexibility for all simulations in the B site (HYD1). Apart from lower occurrence of the hydrophobic features on the slightly longer chain in the center of the ligand, 1QXK simulations show similar behaviour as described for the 1Q1M complex. However the 1QXK ligand is the only one of the three analyzed ligands showing C site interactions. Interestingly, the C site interactions HBD3 and HBD4 of this ligand are among the interactions with highest preference for PTP1B. Additionally, the 1QXK ligand shows several interactions (HBA2, HBA3, HBA4, HBD1 and AR1), all located in the active site binding part of the ligand, where mean occurences in TC-PTP exceed those in PTP1B.



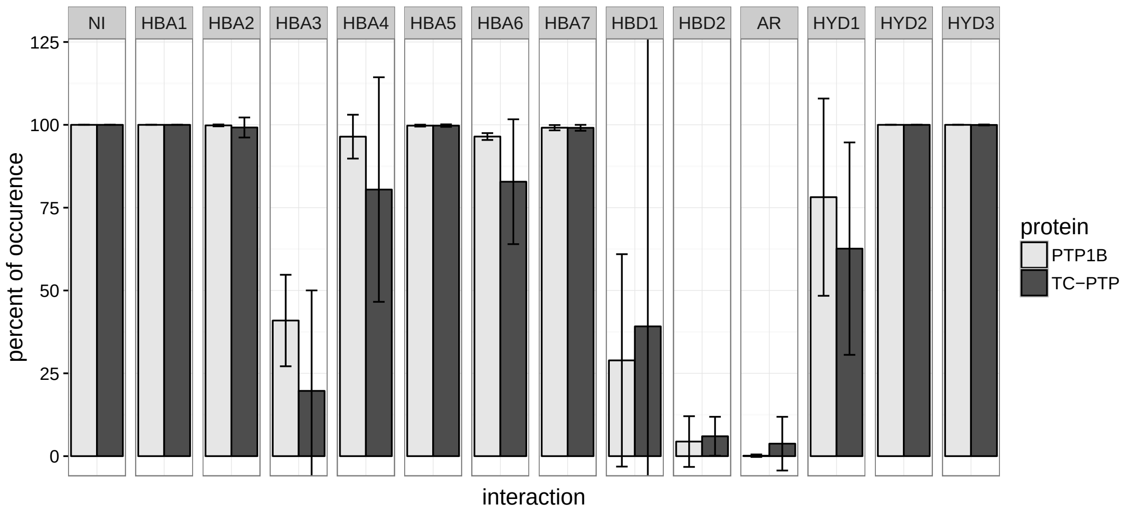
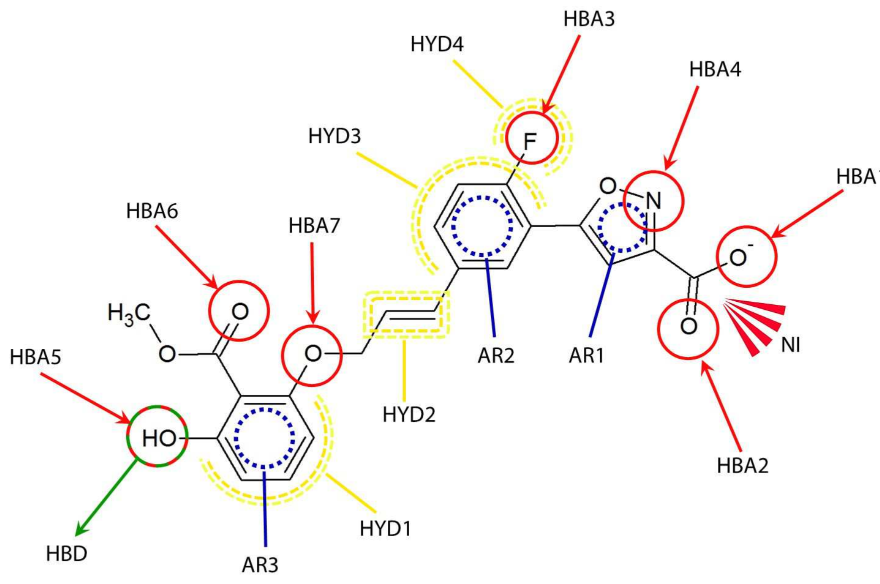


Figure 4.15.: Top: Ligand interactions occurring over the molecular dynamics simulations of the 1XBO complex structures. Bottom: Mean percent of occurrence for the depicted interactions over the three replica of MD simulations of each PTP1B and TC-PTP as classified by the dynophore app in LigandScout. Error bars depict 95% confidence intervals.



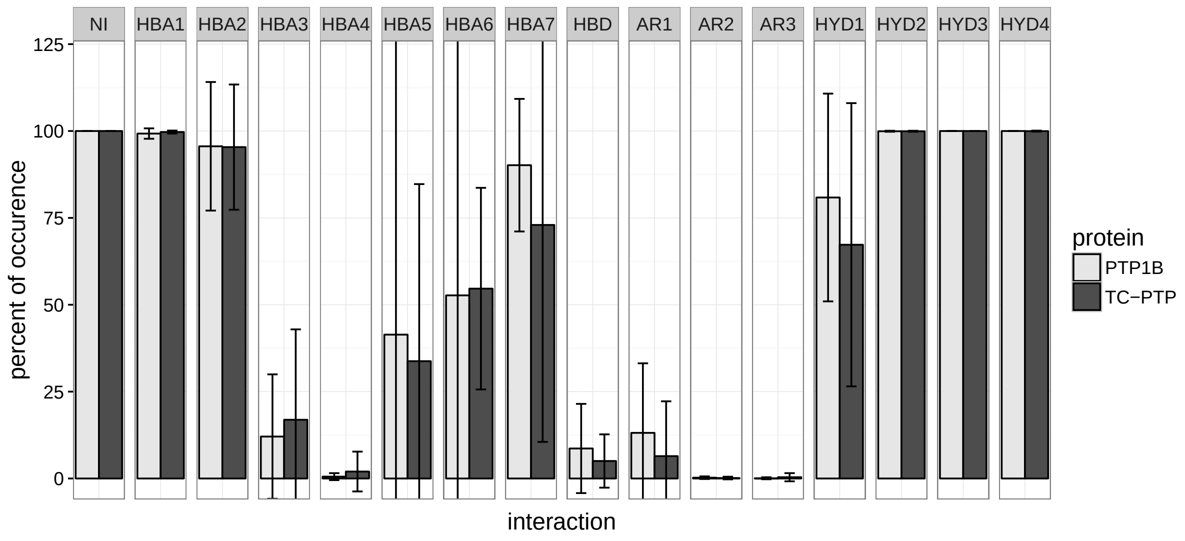


Figure 4.16.: Top: Ligand interactions occurring over the molecular dynamics simulations of the 1Q1M complex structures. Bottom: Mean percent of occurrence for the depicted interactions over the three replica of MD simulations of each PTP1B and TC-PTP as classified by the dynophore app in LigandScout. Error bars depict 95% confidence intervals.

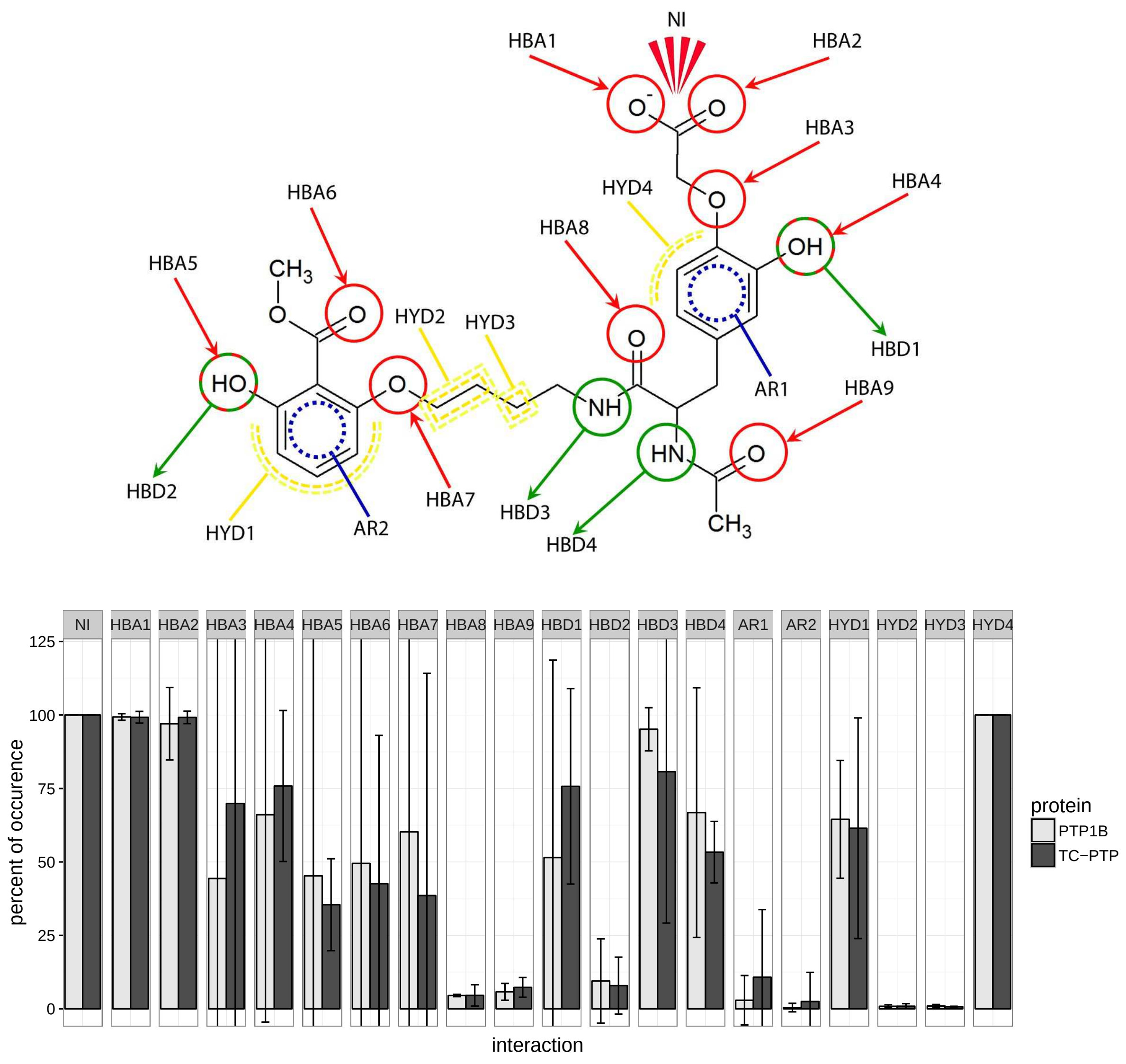


Figure 4.17.: Top: Ligand interactions occurring over the molecular dynamics simulations of the 1QXK complex structures. Bottom: Mean percent of occurrence for the depicted interactions over the three replica of MD simulations of each PTP1B and TC-PTP as classified by the dynophore app in LigandScout. Error bars depict 95% confidence intervals.

4.1.6. Shape Complementarity Analysis

For each of the previously described MD runs, the shape complementarity of the protein binding site and the bound ligand was calculated as described in detail in the next paragraphs: Basis for the shape complementarity are point maps as created by the open source tool POVME2 [94, 95]. Figure 4.18 shows the size and location of the starting grids for the different PTP1B structures in the respective protein structure. To manage computing times, the starting maps were chosen to enclose only the included ligand and the surrounding binding site. This leads to slightly deviating protein areas considered for the shape fit calculations. Their total sizes are similar with 45415 points for 1XBO and 41248 points for 1QXK.

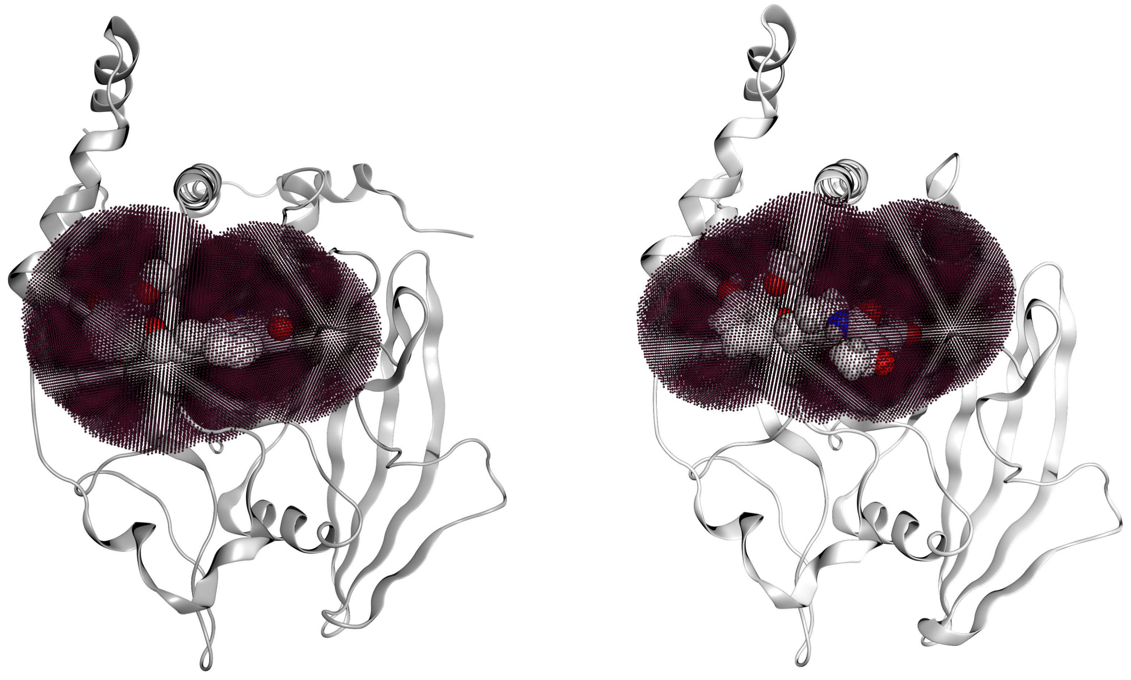


Figure 4.18.: Maps of starting points for shape map calculation for 1XBO and the corresponding TC-PTP homology model (left) and 1QXK and the corrensponding TC-PTP homology model (right). Protein backbones are included for size orientation, ligands in space filling representation for binding site location.

Figure 4.19-A shows a schematic 2D depiction of a grid of points spanning the binding area of a protein as created by the POVME2 tool together with a bound ligand. For every frame of an aligned molecular dynamics trajectory, subsets of this same grid can be created. On the one hand, two maps with all points of distance to ligand heavy atoms of at least 1.59 Å and 0.59 Å were extracted (see Figure 4.19-C and D, respectively). The thresholds were chosen to obtain a point surface of two to three data points of 0.5 Å distance around the ligand. The points are placed around a distance of one van der Waals radius of a hydrogen atom (1.09 Å) to the ligand heavy atoms. Additionally, the POVME2 tool was modified to create a subset of the starting grid of the points inside the protein surface as depicted in Figure 4.19-B. The modified python script can be found in section A.1 of the Appendix. Further processing of the point maps was performed using the software R. The script appended in section A.2 is used to calculate a ligand surface map as the difference of both maps (Figure 4.19-E). This surface map is then used to calculate the overlap of the protein map and the ligand surface map (E) for each frame and converts it to a ratio of the number of overlapping points divided by the total number of ligand surface points for this frame as a parameter of shape complementarity. Employing a nearest neighbor algorithm as implemented as function nn2 from the RANN [111] package in R [89] it also assigns nearest ligand heavy atoms to each ligand surface map point (step E). This enables to calculate an atom related overlap ratio as illustrated in Figure 4.19-G in order to trace back especially good or bad protein-ligand complementarity to a certain part of the ligand.

Plotting of the resulting shape fit ratio of the 1XBO derived complexes together with the calculated strain energy of the ligand reveals interesting tendencies (Figure 4.20):

While the strain energy distributions are almost equal for PTP1B and TC-PTP, the shape fit distributions differ for the two proteins. Mean and median shape fit ratios are higher for PTP1B (0.653 and 0.654, respectively) compared to the TC-PTP values (0.628 and 0.630). Additionally, the shape fit distribution is broader for TC-PTP: the standard deviation is calculated to 0.033 compared to 0.025 for PTP1B. To test whether those observed differences are statistically significant, a Mann-Whitney-U-Test was performed on the data resulting in a p-value below 2.2e-16 for a null hypothesis that the two distributions are equal. Therefore the difference is statistically highly significant. Despite this significant difference in shape fit, there seems to be no difference in ligand strain for both proteins. This could suggest that the ligand does not adopt significantly different conformations, but only the protein shows different flexibility to adapt to those conformations. Hence, ligand RMSF values where calculated over the simulation to further investigate this aspect.

Figure 4.21 shows the selective ligand present in all 1XBO derived simulations together with the atom numbering used in the RMSF (Figure 4.22) and atom-wise shape fit plots (Figures 4.23 to 4.26). The per atom shape fit plots reveal five atoms (C2, C3, C14, O2, O4) with significantly increased shape fit in PTP1B compared to TC-PTP.

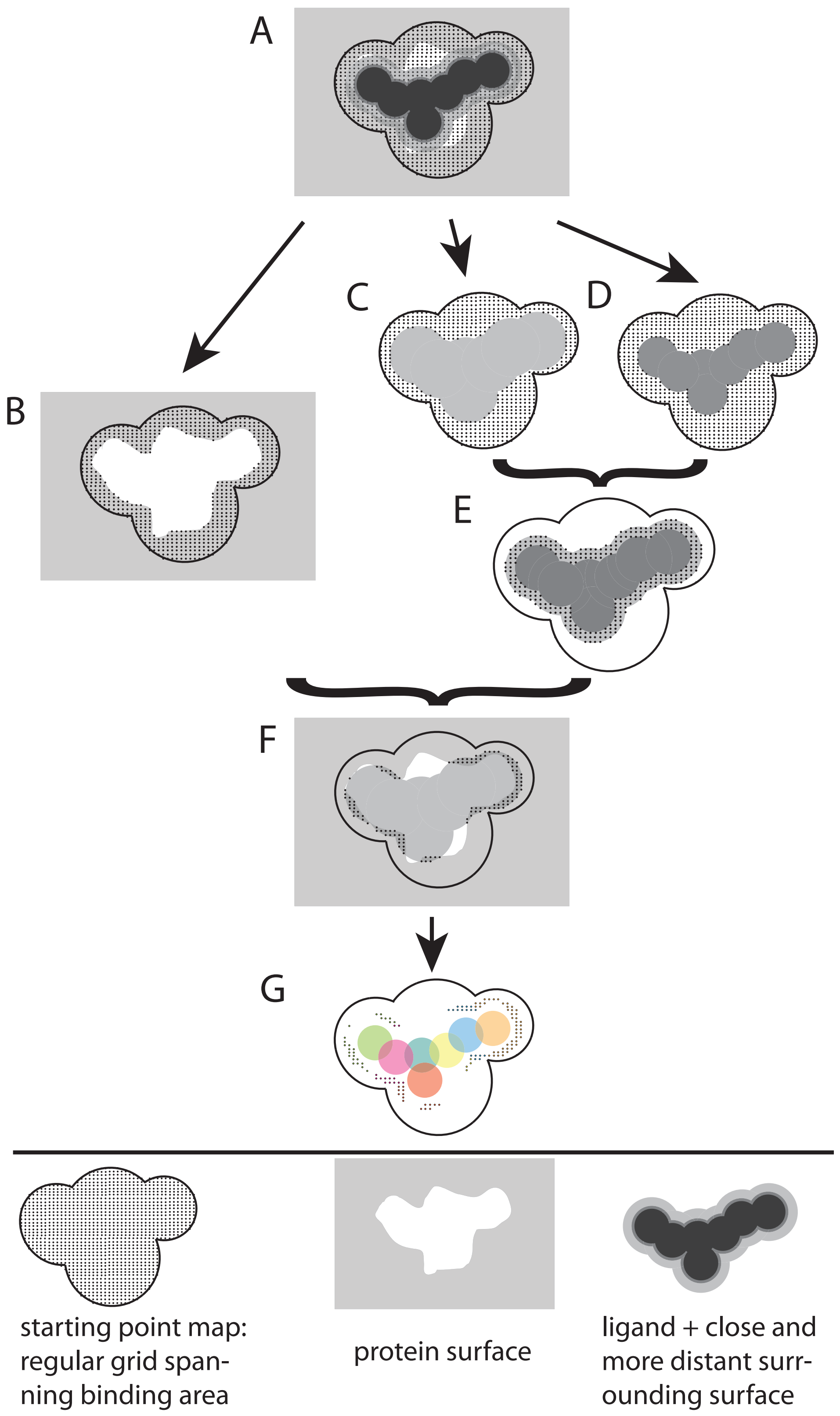

A
B
C
D
E
F
G
starting point map: regular grid span- ning binding area
protein surface
ligand + close and more distant surr- ounding surface

Figure 4.19.: Schematic representation of the shape complementarity workflow.

Further, four atoms (C15, C16, C35, O1) show a slightly better shape fit for the ligand in PTP1B. It is noteworthy that all of the identified atoms except for C35 are located in the catalytic pocket of the protein during simulation and not in the B site as expected. The differences in total ligand shapefit therefore mainly seem related to the catalytic cavity (A site), which is almost identical in amino acid sequence for both proteins. Interestingly, the ligand RMSF plot shows very similar RMSF values of the B site atoms (C32-39 and O11, O13, O14) for both proteins, whereas some atoms (C14-16 and O2) although buried in the active site cavity during simulation for both proteins, show the biggest differences with higher stability in PTP1B. Shape complementarity and ligand RMSF together therefore suggest increased protein-ligand shape fit with resulting higher ligand stability in the active site cavity for PTP1B.

Shapefit calculations were also performed for the PTP1B an TC-PTP complexes based on crystal structure 1QXK (see Figure 4.27). The shape fit distributions reveal the same tendency as for the 1XBO derived complexes: Mean and median shape fit ratios are higher for PTP1B (0.528 and 0.526, respectively) compared to the TC-PTP values (0.506 and 0.502) and the distribution is broader for TC-PTP with standard deviation of 0.033 for PTP1B and 0.046 for TC-PTP. Equivalent to the 1XBO simulations the Mann-Whitney-U-Test was performed and resulted in a p-value below 2.2e-16 for a null hypothesis that the two distributions are equal. The difference in mean of the two distributions is therefore statistically highly significant. Different to the 1XBO simulations, the 1QXK simulations additionally show a difference in the calculated ligand strain distributions. For 3521 of the 12633 frames of PTP1B the ligand strain is below 53, while only 181 frames of TC-PTP show such a low ligand strain. Interestingly, the 2D plot reveals that the low ligand strain values of the PTP1B frames do not correlate with especially low shape fit ratios.

Analogously to the calculations on 1XBO derived simulations, ligand RMSF and atom-wise shape fit were calculated for the 1QXK based simulations to further investigate the relationship between ligand flexibility and shape fit. Figure 4.28 shows the selective ligand present in all 1QXK derived simulations together with the atom numbering used in the RMSF (Figure 4.29) and atom-wise shape fit plots (Figures 4.30 to 4.34). Unexpectedly, the atoms C2, C3, C24, O5, O6 and O7 show a tendency for increased shape fit in TC-PTP despite the overall observation of better shape fit of the ligand to PTP1B. These atoms all belong to the part of the ligand which is bound to the catalytic cavity. The opposite tendency is found for atoms C15, C23, C38, C39, N1, N2, O1, O11 and O12. Five of those atoms (C38, C39, O1, O11 and O12) belong to the ligand part bound to the second phosphotyrosine binding site. The part of the ligand containing N1, N2 and C23 is bound to the YRD-loop at site C of the binding area. The biggest shift in shape fit ratio is found for two of those atoms bound to the C site (C23 and N2). The biggest differences in ligand RMSF are found for the B site ligand atoms C32 to C39 and O1, O11, O12 and O13 while A and C site bound atoms show comparably low RMSF differences in PTP1B compared to TC-PTP. Ligand atoms in the TC-PTP based simulations always show similar or increased flexibility than the same ligand atoms are showing in PTP1B simulations. Combined results of the shape fit calculations together with ligand strain and ligand RMSF could indicate that for this case the B and C site parts are the ones responsible for selectivity. As in the B site part the reduced shape fit in TC-PTP seems to be connected to increased flexibility of the ligand, for the C site part of the ligand this is not observed. However, it is possible that the number of frames with especially low ligand strain in PTP1B is induced by a special conformation of the YRD-loop which is not possible for TC-PTP and which enables good binding to N1 and N2 via hydrogen bonding, while other conformations possible for both PTP1B and TC-PTP lead to a slight increase in ligand strain to preserve the hydrogen bonding interactions to the protein.

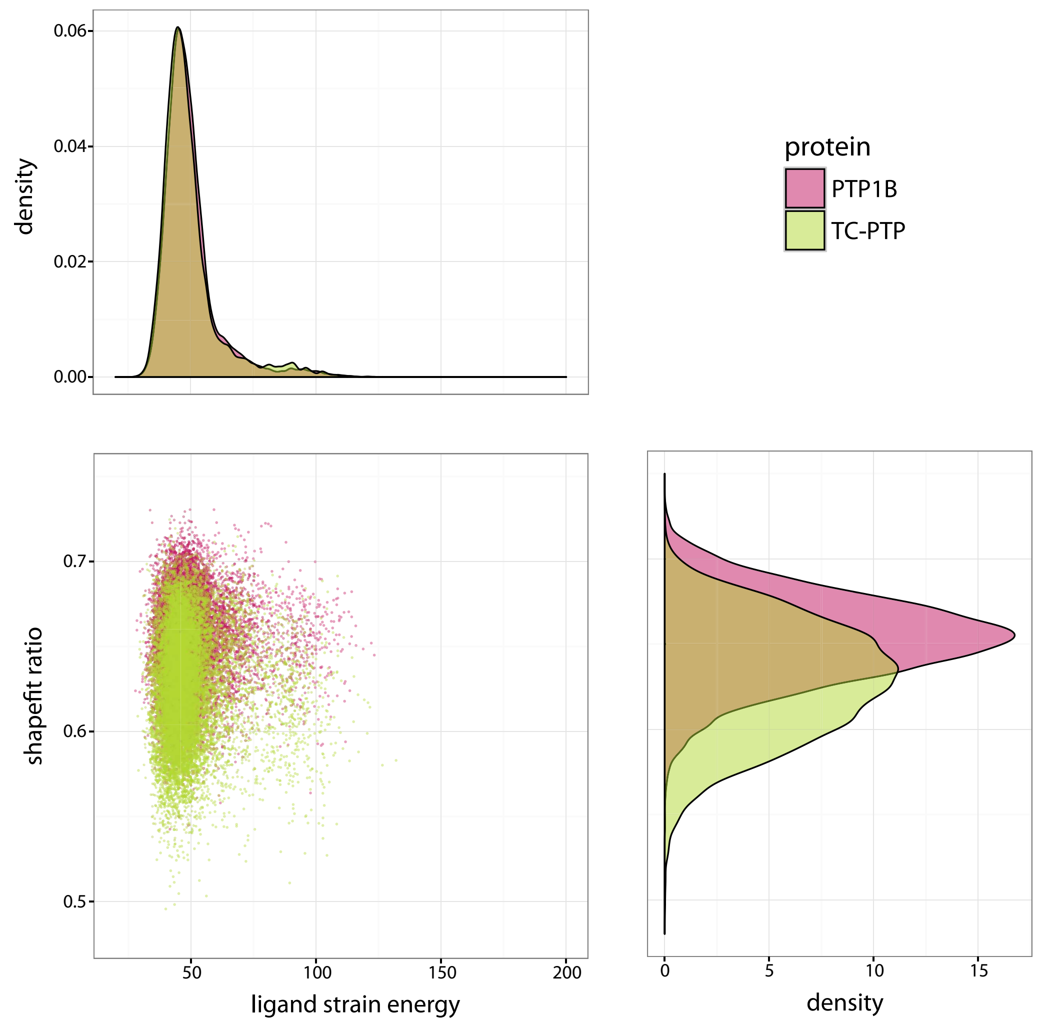


Figure 4.20.: Shapefit ratio and ligand strain energy for all three simulations of both PTP1B and TC-PTP based on the PTP1B crystal structure 1XBO.

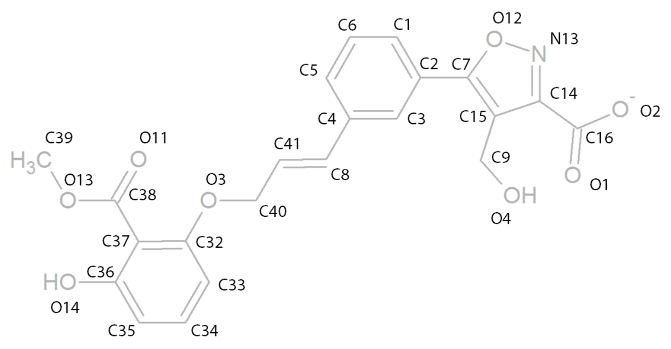


Figure 4.21.: Selective ligand present in the simulations based on crystal structure 1XBO with atom numbering relevant for atom based shape fit plotted below.

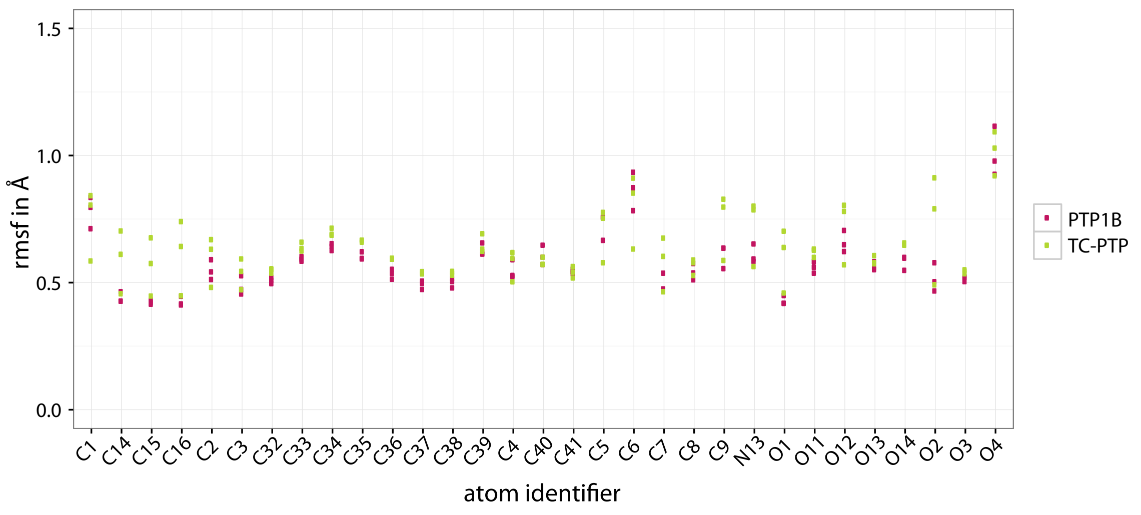


Figure 4.22.: Atom wise ligand RMSF for the PTP1B and TC-PTP simulations based on crystal structure 1XBO.

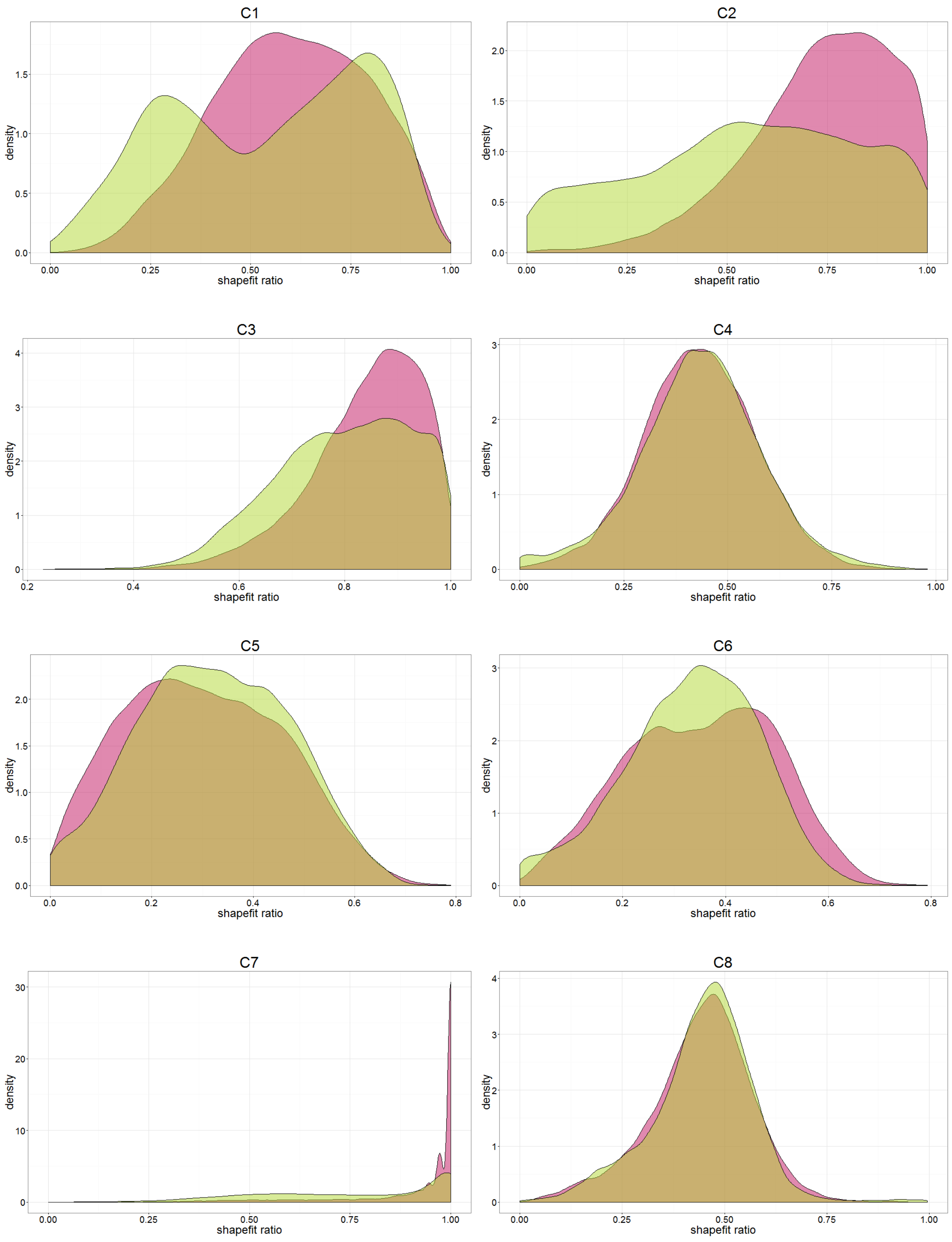


Figure 4.23.: Density plots for atom-wise shape fit ratio in PDB complex 1XBO; pink: PTP1B, green: TC-PTP.

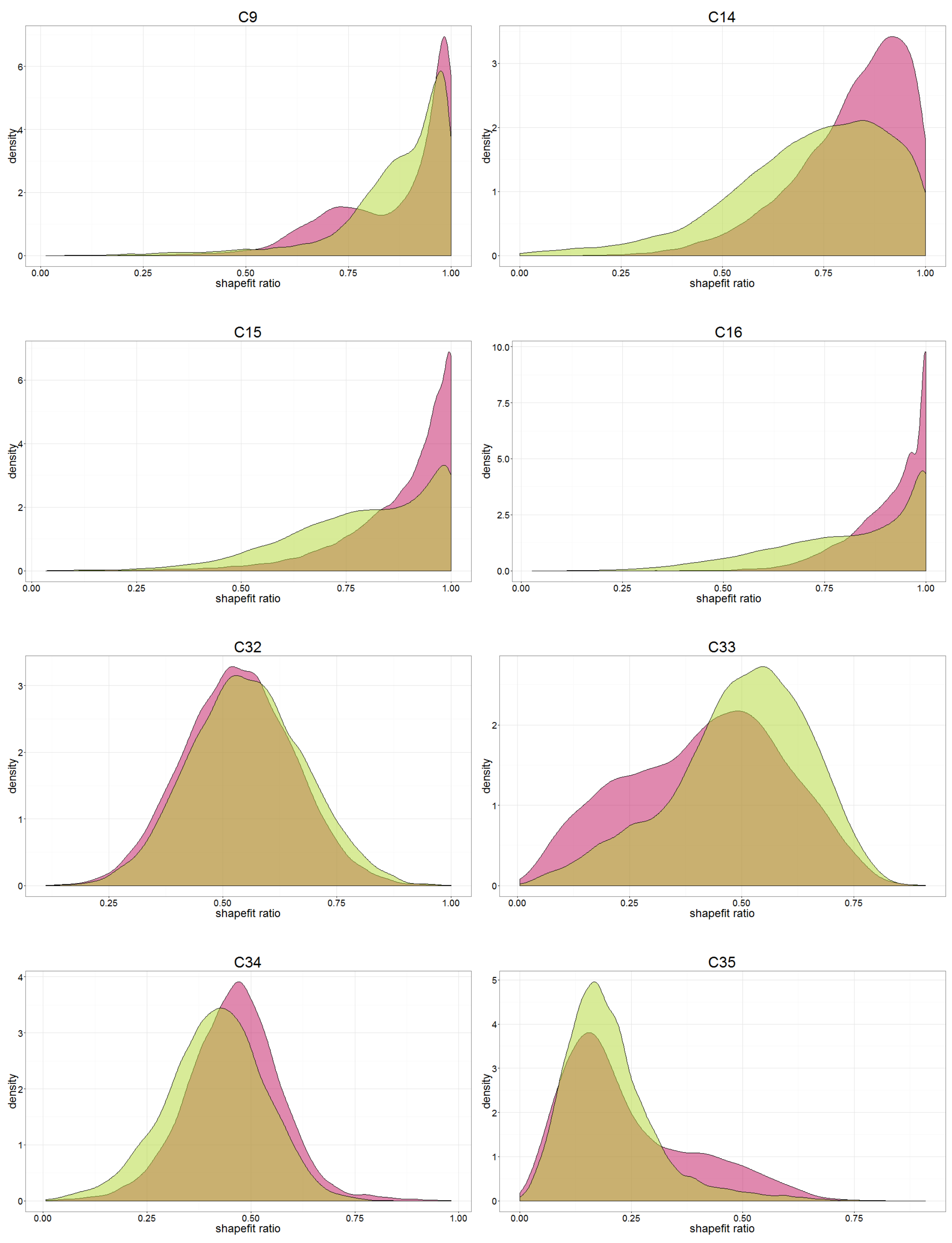


Figure 4.24.: Density plots for atom-wise shape fit ratio in PDB complex 1XBO; pink: PTP1B, green: TC-PTP.

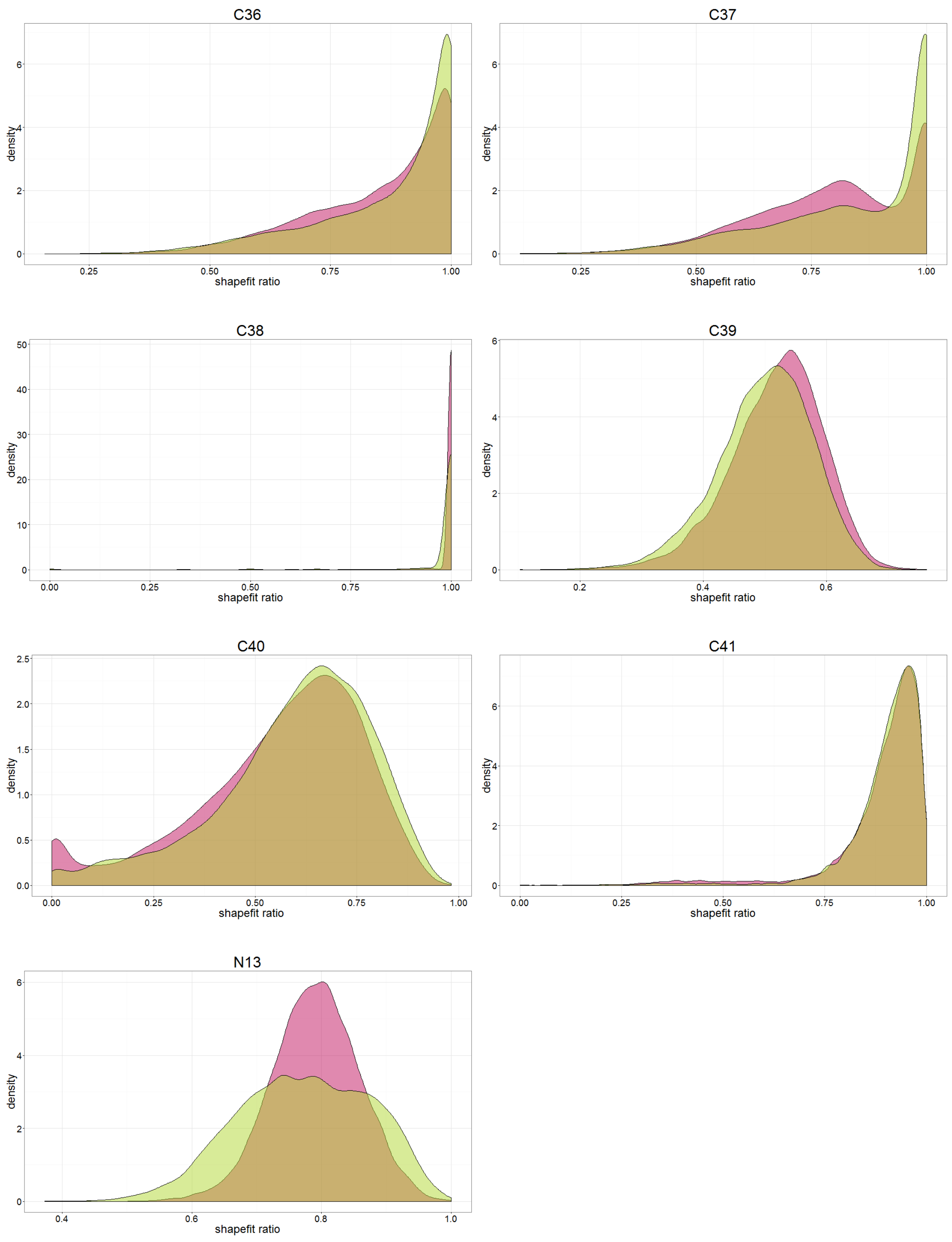


Figure 4.25.: Density plots for atom-wise shape fit ratio in PDB complex 1XBO; pink: PTP1B, green: TC-PTP.

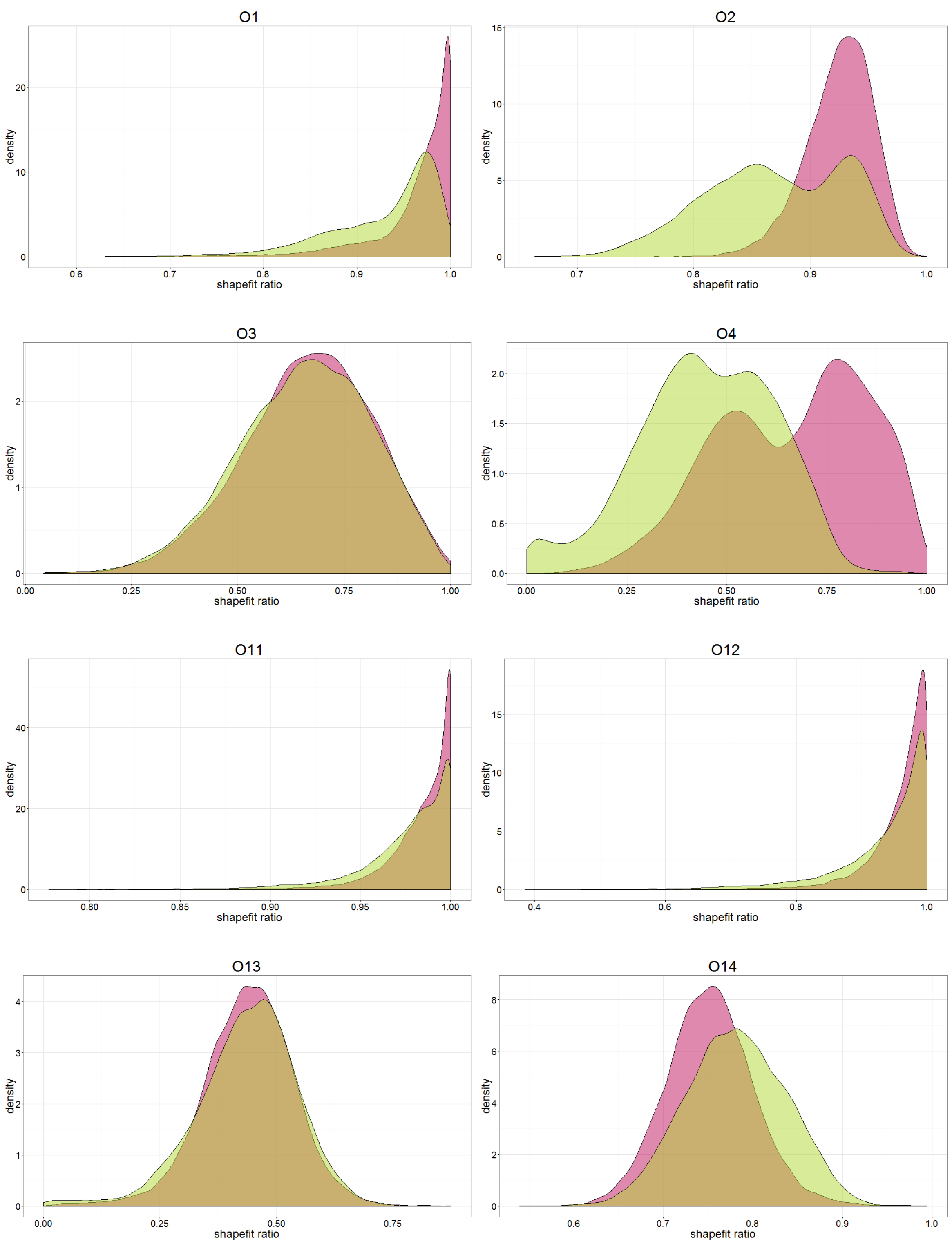


Figure 4.26.: Density plots for atom-wise shape fit ratio in PDB complex 1XBO; pink: PTP1B, green: TC-PTP.

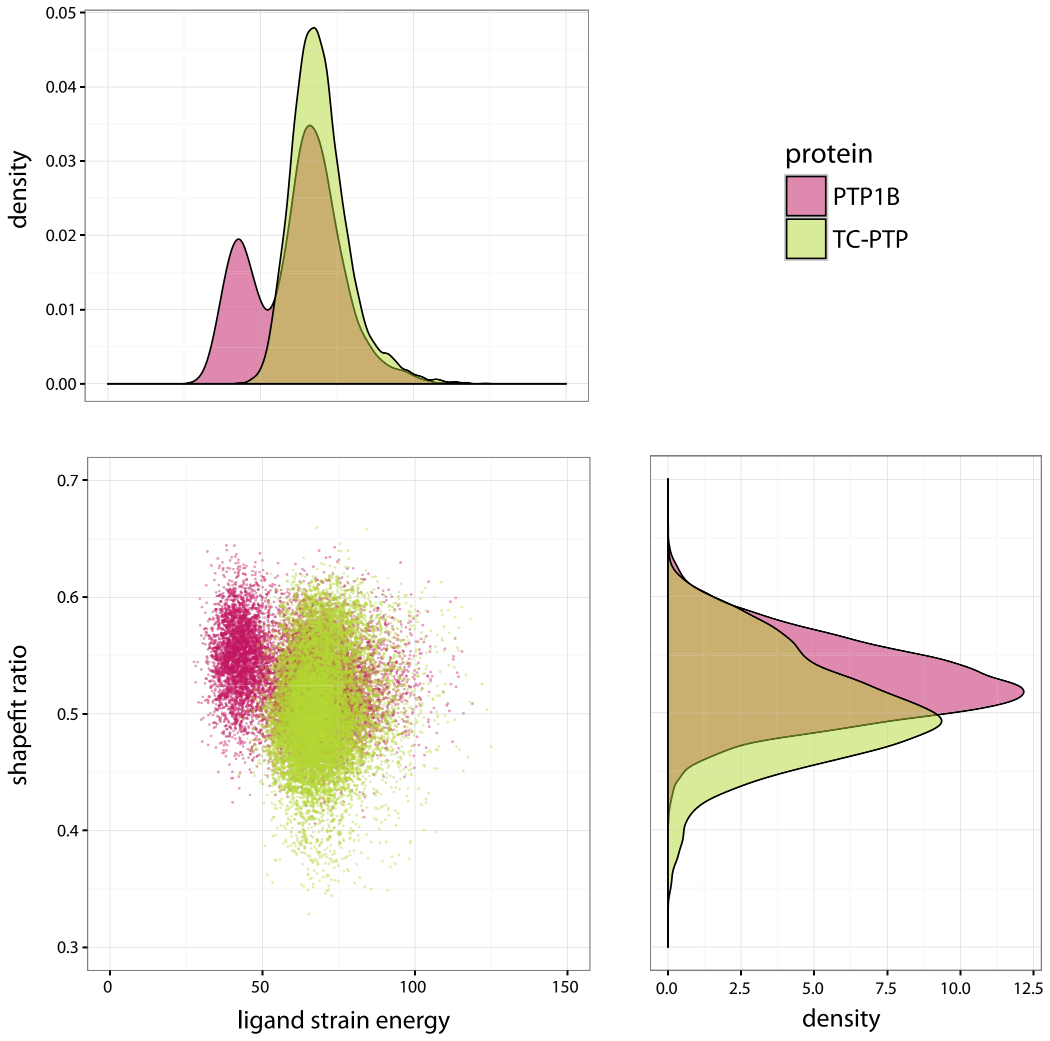


Figure 4.27.: Shapefit ratio and ligand strain energy for all three simulations of both PTP1B and TC-PTP based on the PTP1B crystal structure 1QXK.

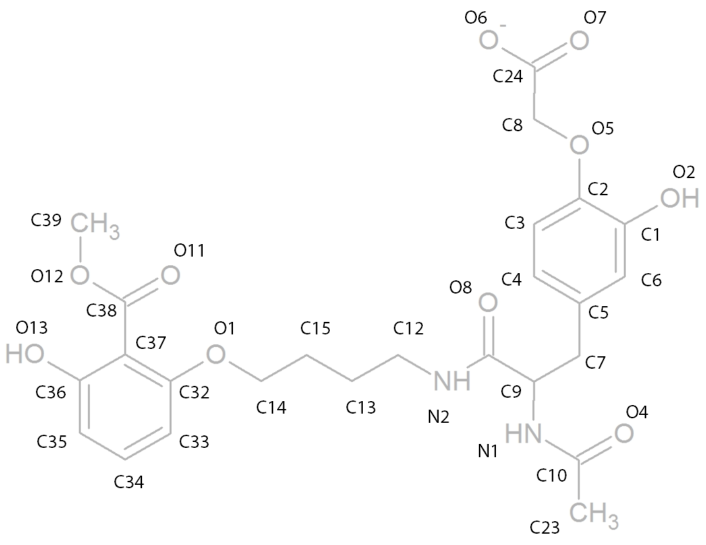


Figure 4.28.: Selective ligand present in the simulations based on crystal structure 1QXK with atom numbering relevant for atom based shape fit plotted below.

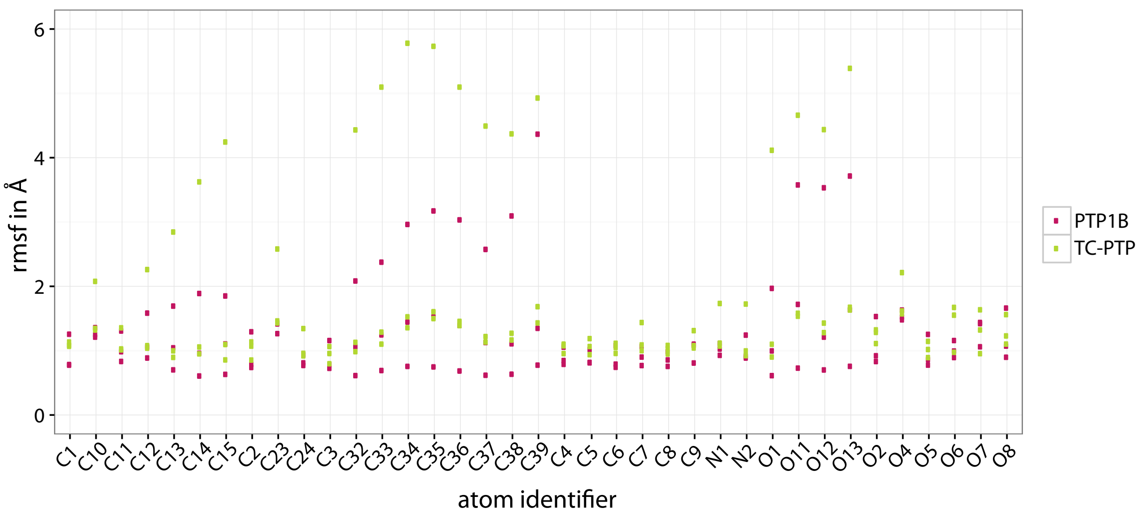


Figure 4.29.: Atom wise ligand RMSF for the PTP1B and TC-PTP simulations based on crystal structure 1QXK.

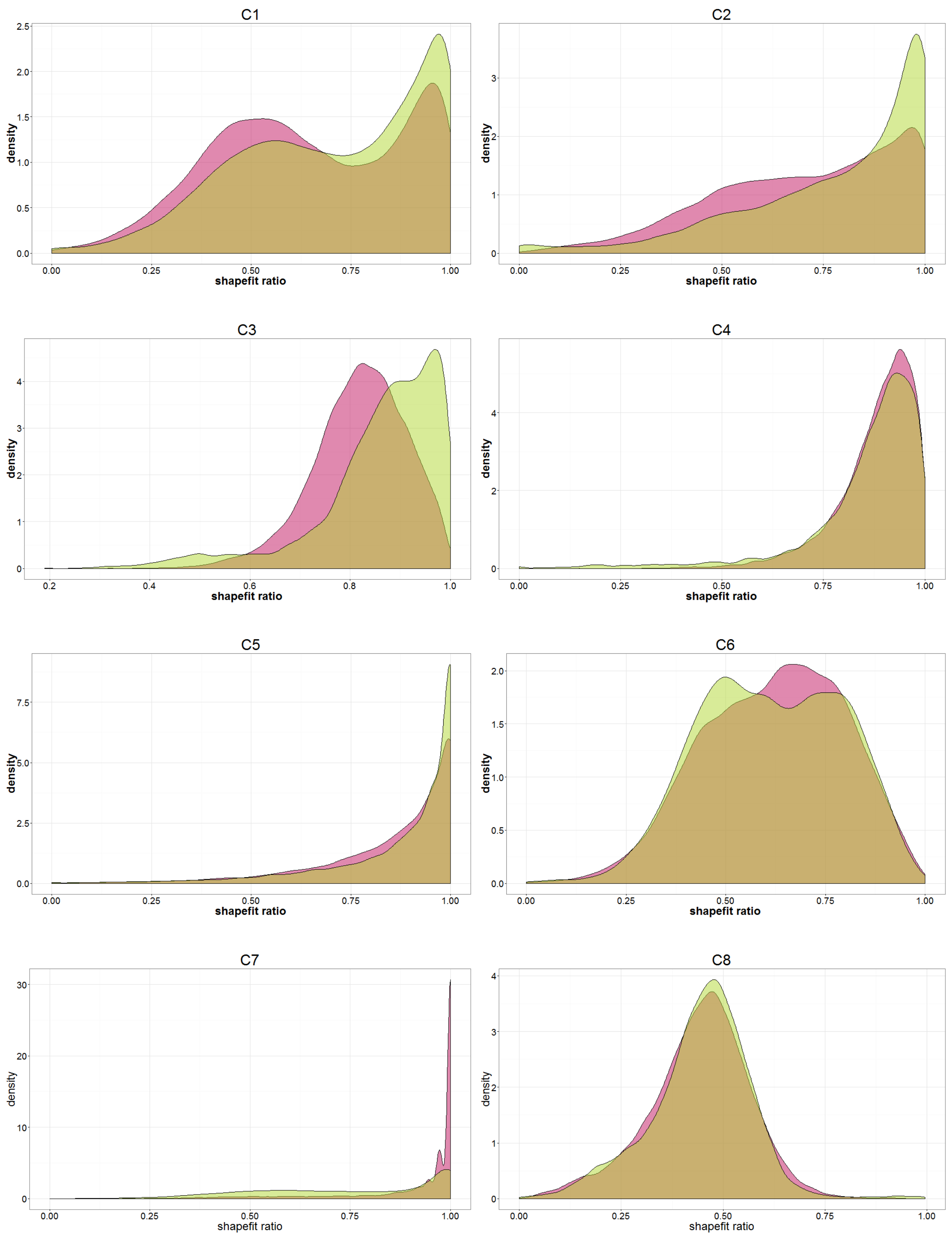


Figure 4.30.: Density plots for atom-wise shape fit ratio in PDB complex 1QXK; pink: PTP1B, green: TC-PTP.

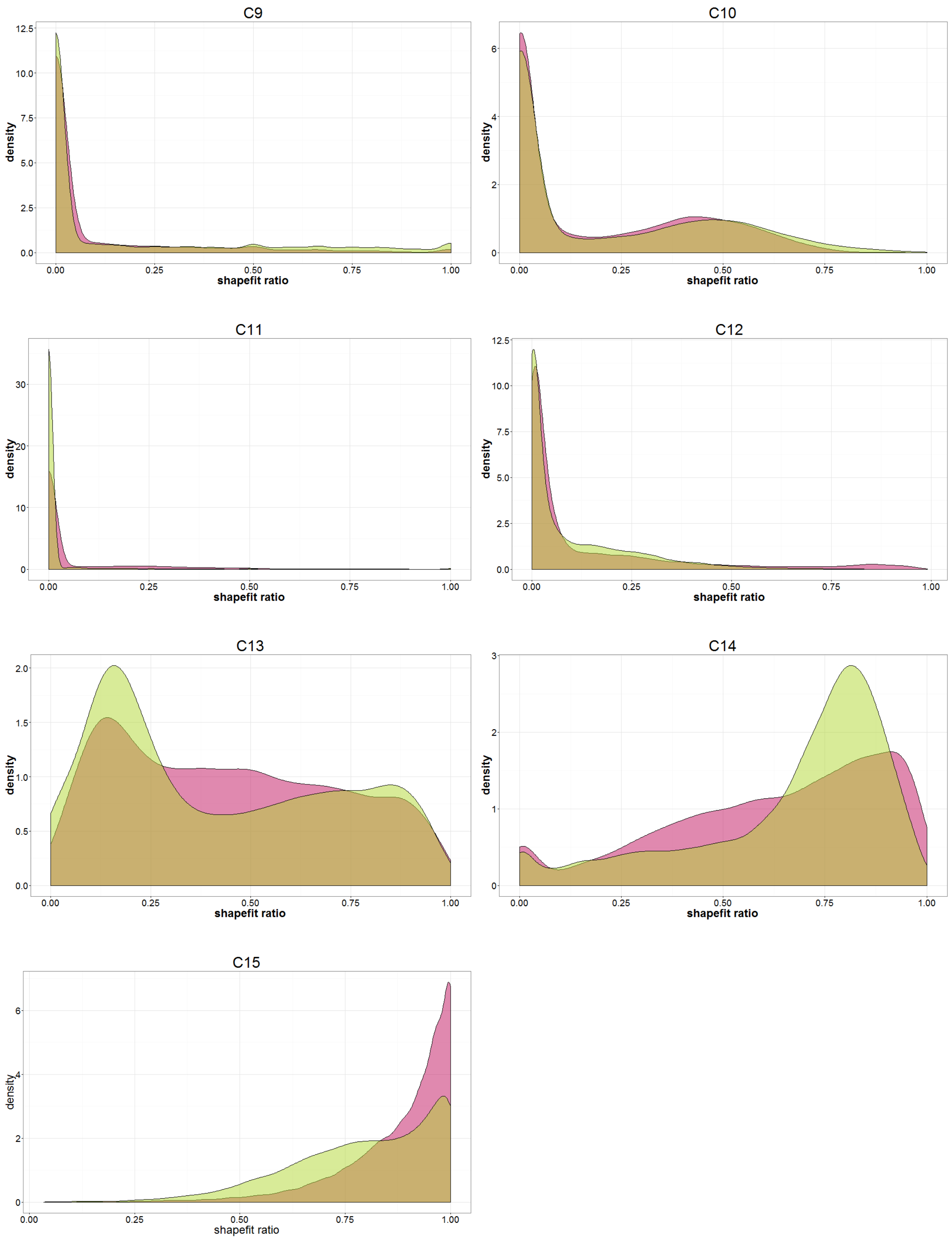


Figure 4.31.: Density plots for atom-wise shape fit ratio in PDB complex 1QXK; pink: PTP1B, green: TC-PTP.

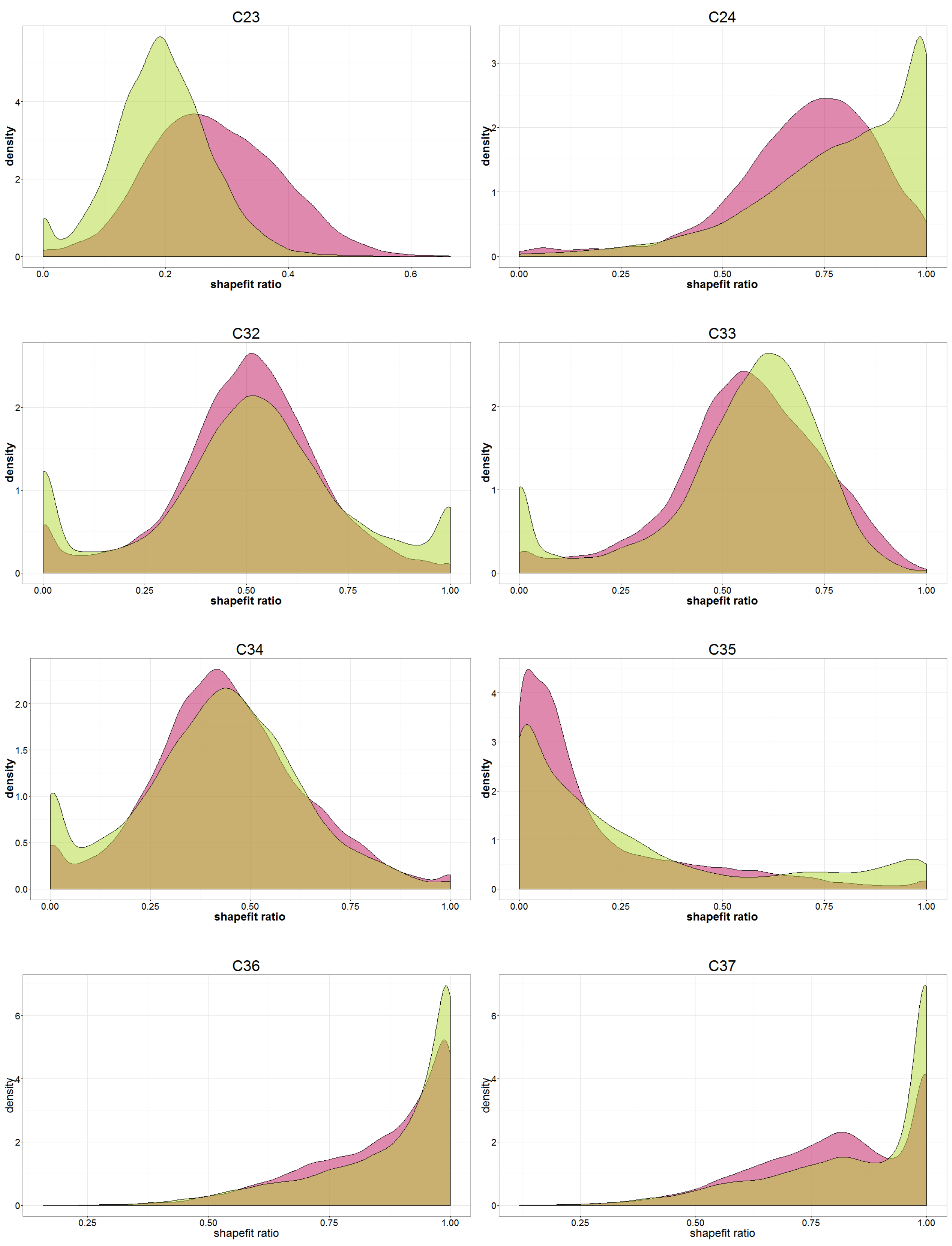


Figure 4.32.: Density plots for atom-wise shape fit ratio in PDB complex 1QXK; pink: PTP1B, green: TC-PTP.

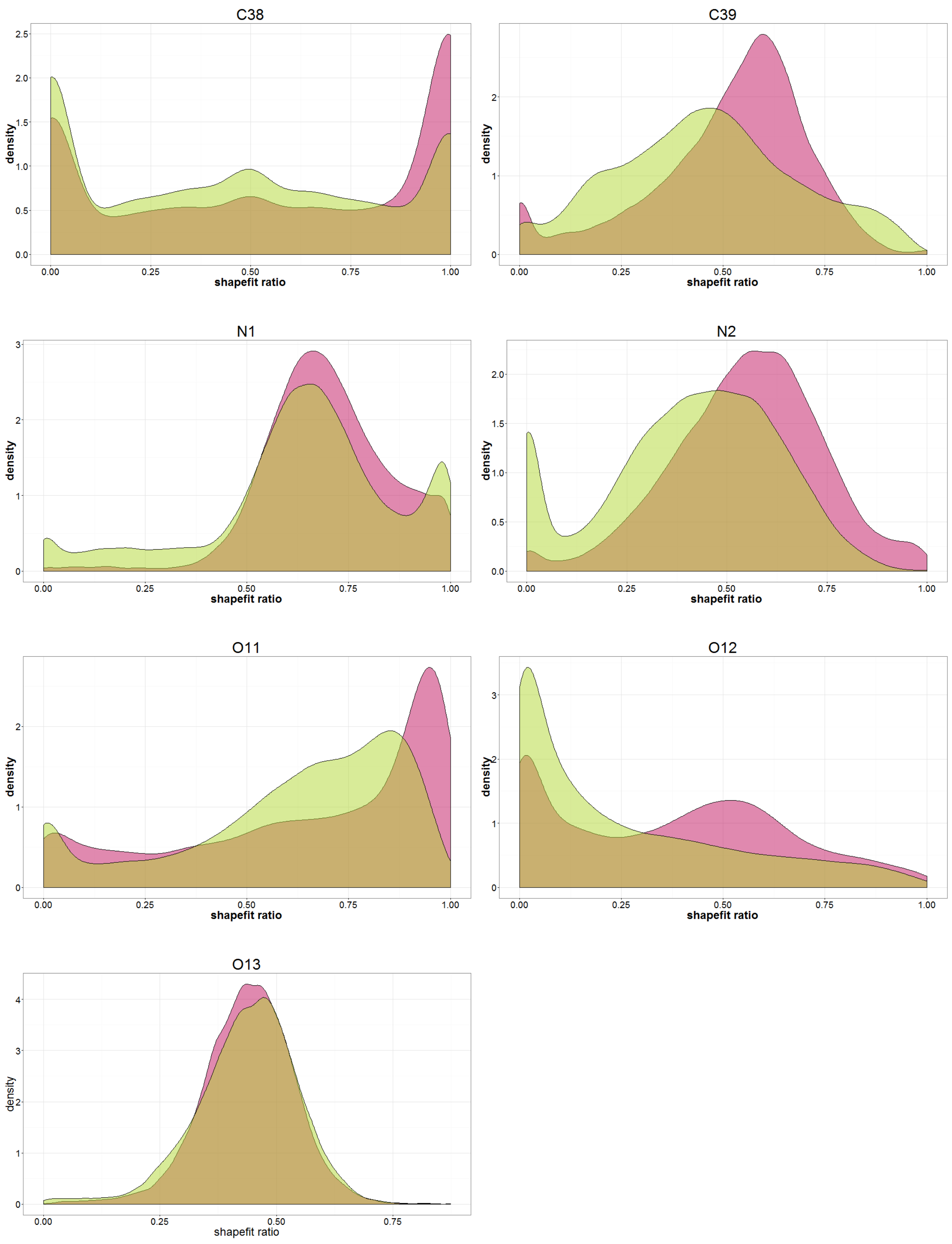


Figure 4.33.: Density plots for atom-wise shape fit ratio in PDB complex 1QXK; pink: PTP1B, green: TC-PTP.

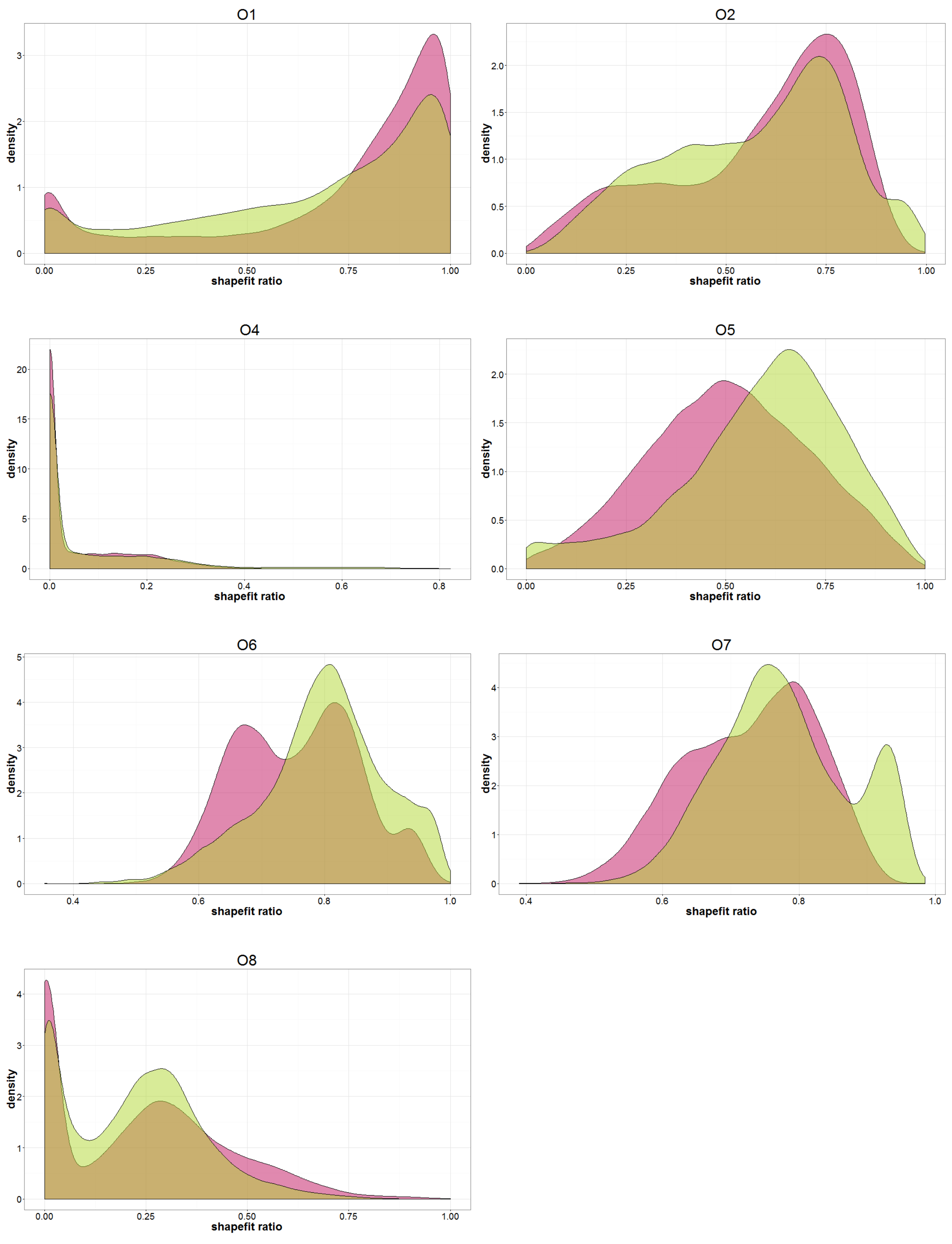


Figure 4.34.: Density plots for atom-wise shape fit ratio in PDB complex 1QXK; pink: PTP1B, green: TC-PTP.

4.2. Part II: Binding Site Shape Clustering and Screening

4.2.1. Generation of Input Data

Due to the observation that stronger inhibitors of PTP1B are found to preferably bind to the closed conformation of the active site cavity this approach strives to discover PTP1B selective compounds targeting the closed active site conformation. In order to prevent the occurrence of loop opening events during the simulations and potentially connected conformational changes in other parts of the protein, the protein was guided to keep the closed conformation with a phosphotyrosine residue bound to the catalytic cavity. As starting system for PTP1B, PDB structure 1PTY was chosen, which contains phosphotyrosine residues in the catalytic cavity and the second phosphotyrosine binding site. The second phosphotyrosine was removed prior to the molecular dynamics simulations, with the aim not to unnecessarily bias the rest of the binding cavity. Since a closed cavity complex was not available for TC-PTP, a homology model was build based on the PTP1B complex 1PTY. To yield a structure equivalent to the PTP1B structure for simulation, the coordinates of the catalytic phosphotyrosine and the adjacent catalytic water molecule were transposed to the TC-PTP homology model. Molecular dynamics simulations were performed on both complexes in triplicates over 50 ns each. Analogously as for the previously described molecular dynamics simulations C_α-RMSD and RMSF plots were created and analyzed.

Similar to the previously described complexes, the simulations of the 1PTY derived complexes show stable system behavior with RMSD values below 2.4 Å. RMSD plots in Figure 4.35 show that the PTP1B simulations reach equilibrium after about 5 ns, while the TC-PTP structures still seem not to have finished the rearrangement after 10 to 15 ns. Further they show a slightly higher overall RMSD than the PTP1B simulations. Both observations are in good agreement with our expectations due to the fact that ligand coordinates were transposed from the PTP1B structure without energy minimization. The time point of 15 ns was therefore considered as starting point of the simulations for further analysis.

Starting from the determined point of equilibration, point maps depicting the binding site shape were generated for each frame with the tool POVME2 [94, 95]. Again the C_α-atoms of the stable active site loop amino acids Cys215-Arg221 were chosen for alignment of all frames. The TC-PTP trajectories were aligned on the first frame of the PTP1B trajectories to enable direct comparison of the resulting data due to use of the same starting point map for the POVME2 calculations. The starting point map to-

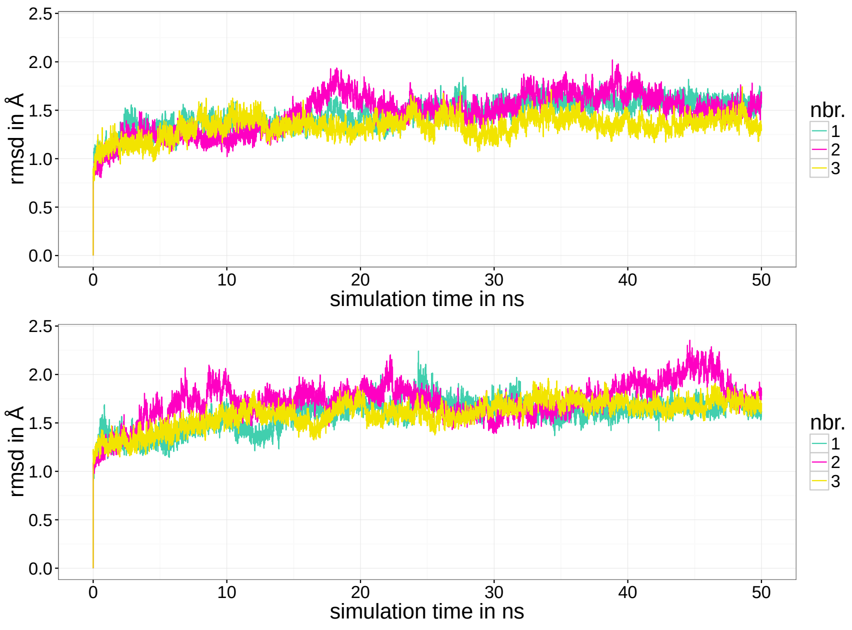


Figure 4.35.: RMSD plots of the molecular dynamics simulations performed on the 1PTY crystal structure with a cropped phosphotyrosine in the catalytic pocket (top) and the corresponding TC-PTP homology model complex (bottom).

gether with the protein surface of the starting frame is depicted in Figure 4.37(left). The starting point map spans the whole binding area of PTP1B with all subsites (compare Figure 4.1). However, the outside of the lid closing the catalytic pocket is only partially covered. Considerations on the influence of this fact regarding the outcome of the clustering workflow are included in the following chapter.

4.2.2. Bootstrapping and Clustering

Since the amount of storage capacity required for clustering the whole dataset exceeded the available resources, bootstrapping was used to create three smaller samples of 14598 (one third of the considered frames) frames each [112]. Calculations were then performed on all three samples. Clustering results for each sample were further processed with an R script, which can be found in section A.3 to guide the selection of the appropriate clustering level for further analysis. The approach developed here is similar to the elbow method to find the optimal number of clusters for a dataset [103]. Like the elbow method, it strives to assess how intra-cluster variance changes for each step and especially after which step it does not increase significantly anymore. Due to the nature of clustering algorithms, clusters will get more uniform the smaller they get, but too small clusters will result in overfitting or unnatural segregation. In this analysis, only clusters that could clearly be assigned to one of the two proteins PTP1B or TC-PTP were considered, since bigger mixed clusters were considered more likely to be less uniform and also less important in order to find differences between the proteins. As clearly assignable to one protein, all clusters were considered that contain over 92.5% PTP1B or TC-PTP frames.

In order to get a parameter for intra-cluster variance, for the clustering steps 1 to 50, for each cluster the ratio of occupancy of every point of the starting point map was calculated. From those occupancies, the ratio of the always present points (occupancy $> 95\%$) and always absent points (occupancy $< 5\%$) from the whole starting point map was computed, which represents a measure of the uniformness of each cluster. This uniformness parameter was then plotted for the considered clusters (see Figure 4.36) together with the mean value over all clusters with curves delimiting the ± 1 standard deviation area around the mean.

All plots show a steep increase of the intra-cluster uniformness in the first clustering steps. However, after clustering step 8 for sample three or clustering step 9 for sample one and two, the intra-cluster uniformness is only slightly increasing over the next 40 clustering steps. This bend in the curve signalizes that after clustering step 8 or 9, respectively, the clustering quality does not increase significantly and that this bend is most likely to represent the natural clustering structure of the data well. To continue the work with similar cluster sizes, a common clustering step - step 9 - was chosen for further analysis for all three samples.

Tables 4.3, 4.4 and 4.5 show the results of the clustering at step 9 for the three samples together with the assignment of each cluster to one of the two proteins according to the ratio of frames belonging to each protein. Surprisingly, all clusters at the chosen clustering step match the chosen threshold criterion and can therefore be assigned to one protein. For all samples, the clustering shows a distribution with nice cluster sizes of over 800 frames per cluster and minimal ratios of over 94% of frames belonging to one protein. This suggests that there are indeed differences in the binding site shape of both proteins and that the chosen clustering method seems able to detect these differences well.

Based on this cluster assignment to the two proteins, calculations were performed on all samples in order to find the PTP1B assigned cluster of each sample that is most different to all TC-PTP assigned clusters. With this aim, for each PTP1B cluster the squared occupancy differences of each grid point to each TC-PTP assigned cluster were

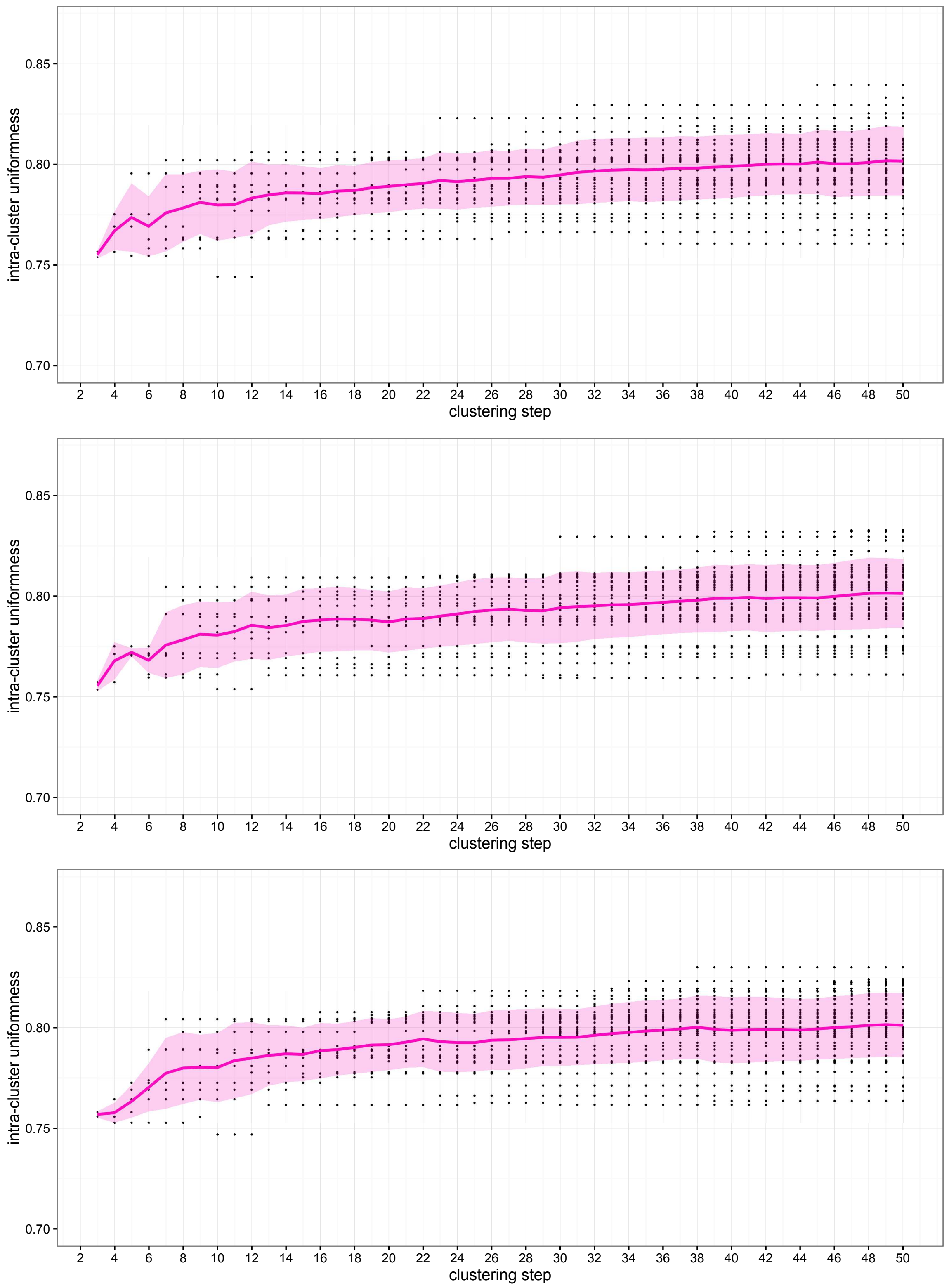


Figure 4.36.: Intra-cluster uniformness over clustering step number for all three samples. Mean value over all clusters per step as pink line; pink area represents ±1 standard deviation area around the mean.

calculated and summed up. Squared differences were chosen to give less emphasis to smaller occupancy differences that might not be statistically significant. The PTP1B cluster with the highest sum was considered to be the most different to all TC-PTP clusters. In the clustering tables (Table 4.3 to 4.5) those clusters are marked with an asterisk.

Cluster Size	*PTP1B Frames*	*PTP1B %*	*TC-PTP Frames*	*TC-PTP %*	*Assignment*
1835	1835	100.0	0	0.0	PTP1B
1044	1044	100.0	0	0.0	PTP1B
2963	93	3.1	2870	96.9	TC-PTP
2049	2049	100.0	0	0.0	PTP1B *
2302	2302	100.0	0	0.0	PTP1B
1163	0	0.0	1163	100.0	TC-PTP
977	1	0.1	976	99.9	TC-PTP
989	0	0.0	989	100.0	TC-PTP
1276	0	0.0	1276	100.0	TC-PTP

Table 4.3.: Clustering of sample 1 at the chosen clustering step. Asterisk marks PTP1B cluster most different to the TC-PTP assigned clusters.

Cluster Size	*PTP1B Frames*	*PTP1B %*	*TC-PTP Frames*	*TC-PTP %*	*Assignment*
2986	1	0.0	2985	100.0	TC-PTP
1775	1775	100.0	0	0.0	PTP1B*
2550	2410	94.5	140	5.5	PTP1B
1885	1885	100.0	0	0.0	PTP1B
939	0	0.0	939	100.0	TC-PTP
970	0	0.0	970	100.0	TC-PTP
1206	1206	100.0	0	0.0	PTP1B
1468	0	0.0	1468	100.0	TC-PTP
819	0	0.0	819	100.0	TC-PTP

Table 4.4.: Clustering of sample 2 at the chosen clustering step. Asterisk marks PTP1B cluster most different to the TC-PTP assigned clusters.

Cluster Size	*PTP1B Frames*	*PTP1B %*	*TC-PTP Frames*	*TC-PTP %*	*Assignment*
2042	2042	100.0	0	0.0	PTP1B*
2886	96	3.3	2790	96.7	TC-PTP
929	0	0.0	929	100.0	TC-PTP
2484	2363	95.1	121	4.9	PTP1B
1505	0	0.0	1505	100.0	TC-PTP
940	0	0.0	940	100.0	TC-PTP
1666	1666	100.0	0	0.0	PTP1B
1139	1139	100.0	0	0.0	PTP1B
1007	0	0.0	1007	100.0	TC-PTP

Table 4.5.: Clustering of sample 3 at the chosen clustering step. Asterisk marks PTP1B cluster most different to the TC-PTP assigned clusters.

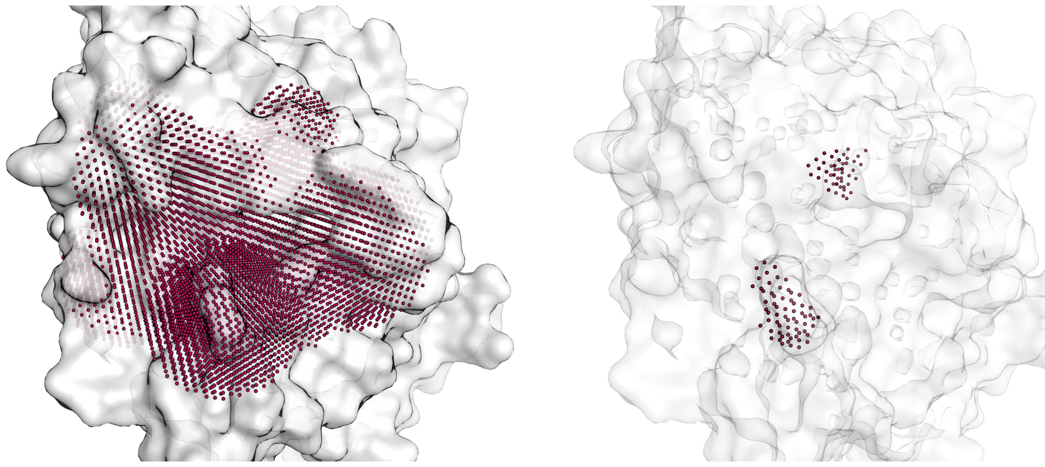


Figure 4.37.: Left: Point map spanning the binding area of PTP1B used as basis to derive protein shape information for clustering together with surface of the first PTP1B frame. Right: Subsection of the points related to selectivity.

4.2.3. Selectivity and Interaction Patterns

Consecutively, all grid points were stored that show at least 70% occupancy difference of the selected PTP1B cluster to all TC-PTP clusters simultaneously. For sample 1 this list comprises 124 points that are part of the available space for ligand binding in TC-PTP, but not in PTP1B. Points that are part of the binding site for PTP1B, but not TC-PTP were not found. Sample 2 shows a similar result with 121 points available for ligand binding in TC-PTP, but not PTP1B and no common points for all clusters with the opposite behavior. Similarly, sample 3 shows 116 points available for ligand binding in TC-PTP, but not PTP1B and no common points for all clusters with the opposite behavior. Interestingly, of the points that are part of the available space for ligand binding for TC-PTP, but not PTP1B, the three samples share 110 points, therefore showing significant overlap of the results. Those shared points are depicted in Figure 4.37(right). The points concentrate on site C of the PTP1B binding area and the upper lid of site A. At the lid of site A, differences in the torsional freedom of the Phe182 sidechain seem responsible for the observed binding site preferences. The points found in the C site span the area of the sidechains of Arg47 and Asp48, where TC-PTP seems to prefer more outward faced conformations.

Screening of all 43754 frames considered in the cluster analysis (21877 for each PTP1B and TC-PTP) with the 110 points correlated with selectivity and also with the subset of 64 points in the vicinity of Arg47 was conducted to reassure their use for distinguishing

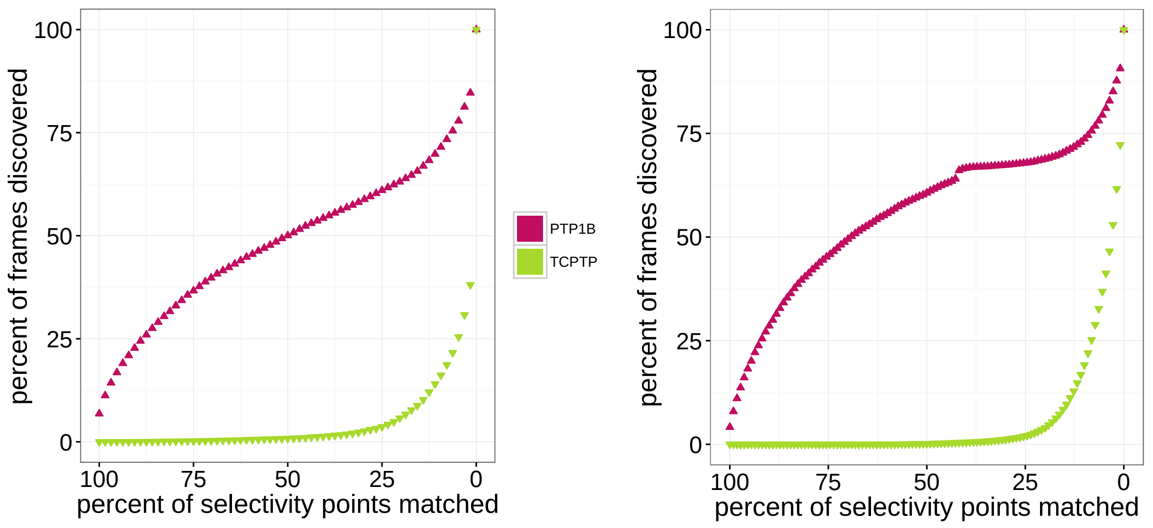


Figure 4.38.: Ratio of frames matching at least x percent of the selectivity map. Left: For the subset of 64 points around Arg47. Right: For the whole map of 110 points.

between PTP1B and TC-PTP frames, but also to select PTP1B frames for the consequent virtual screening workflow. The results are plotted in Figure 4.38. The selectivity map containing all 110 points is fulfilled by 905 PTP1B frames, but no TC-PTP frame. The best fitting frame of all TC-PTP frames matches 73 of the 110 points. The subset of the selectivity map containing the 64 points around Arg47 is fulfilled by 1491 PTP1B frames, but no TC-PTP frame. The best fitting frame of all TC-PTP frames matches 59 of those 64 points. However, many of the TC-PTP frames - 13542 - match none of the 64 points. Further it can be deduced from Figure 4.38 that the TC-PTP frames matching higher numbers of the selectivity map points are rare for both the full map and the subset of 64 points.

From the 905 PTP1B frames matching all of the 110 points of the selectivity map, five diverse frames were extracted for the virtual screening workflow:

- "1PTYorigFrame3290"
- "1PTYrep2Frame5561"
- "1PTYorigFrame5079"
- "1PTYorigFrame7139"
- "1PTYrep2Frame7181"

4.2.4. Virtual Screening

With the aim to find ligands showing selective inhibition of PTP1B, a virtual screening workflow was developed using the previously described selectivity map. In order to add a filtering step to the screening process to increase the likelihood of activity against PTP1B an approach based on molecular interaction fields was chosen to deduce pharmacophoric features from the chosen protein frames, since a purely shape-based approach would lead to many false positive hits with no suitable distribution of pharmacophoric features. A schematic representation of the workflow is depicted in Figure 4.39.

In a first step (Fig. 4.39-A), molecular interaction fields were generated with MOE using probes for hydrogen bond donor, hydrogen bond acceptor and hydrophobic interactions. To translate the interaction energy maxima to pharmacophore feature points, the generated molecular interaction fields were saved in .cns format and further processed using an R script created for this purpose. The script reads the interaction fields and finds interaction energy maxima by comparing the energy at each grid point with all neighboring points. A binding site center is defined and all points more than 15 Å away from this center are excluded. Exclusion volumes are added on those points of the 110 points correlated with selectivity, which are close to the binding site of the considered frame and not inside the protein surrounded by other exclusion volumes (Fig. 4.39-C). This limitation of the exclusion volumes to the necessary ones saves time during screening. Both feature points as well as exclusion volume spheres of each frame are written to a LigandScout pharmacophore format for visual inspection. Upon visual inspection features with high overlap were merged and features distant from the catalytic site were removed. Additionally, less restrictive exclusion volumes were added manually at protein atom positions in binding site regions not covered by the selectivity map derived exclusion volumes (Fig. 4.39-D).

The final models are named by their molecular dynamics frame of origin. Mod_3290 is depicted in Figure 4.39-E. It comprises one hydrogen bond donor feature and four hydrogen bond acceptor features of which one shows high overlap with the hydrogen bond donor feature. The other three hydrogen bond acceptor features are arranged with low overlap in the catalytic cavity. Further the model contains one hydrophobic feature of increased size between the triple of hydrogen bond acceptors and the overlapping hydrogen bond donor / hydrogen bond acceptor feature. It also contains over 50 exclusion volumes of different size in total. The other models show a similar number and arrangement of exclusion volumes and similar feature arrangements as depicted in Figure 4.40: Mod_5079 shows a triple of hydrogen bond acceptor features separated from a fourth hydrogen bond acceptor feature by two hydrophobic features. In mod_5561 a quadruple of hydrogen bond acceptors is separated from a fifth hydrogen bond acceptor by only one hydrophobic feature. Mod_7139 only shows a triple of hydrogen bond acceptor features and two hydrophobic features. In mod_7181 a triple of hydrogen bond acceptor features is joined by a hydrogen bond donor and a hydrogen bond acceptor feature of high overlap and separated from another hydrogen bond acceptor feature by a hydrophobic feature.

All pharmacophore models show high similarity of the feature types and their distribution compared to a PTP1B bound phosphotyrosine residue: They all show at least three overlapping hydrogen bond acceptor features in the catalytic pocket together with a hydrophobic feature in a distance of 4 to 5 Å to the nested hydrogen bond acceptor features, which are also found for phosphotyrosine and other PTP1B ligands. However, they all show additional features close to the YRD loop to ensure interactions with residues of the C site in a selective binding site conformation.

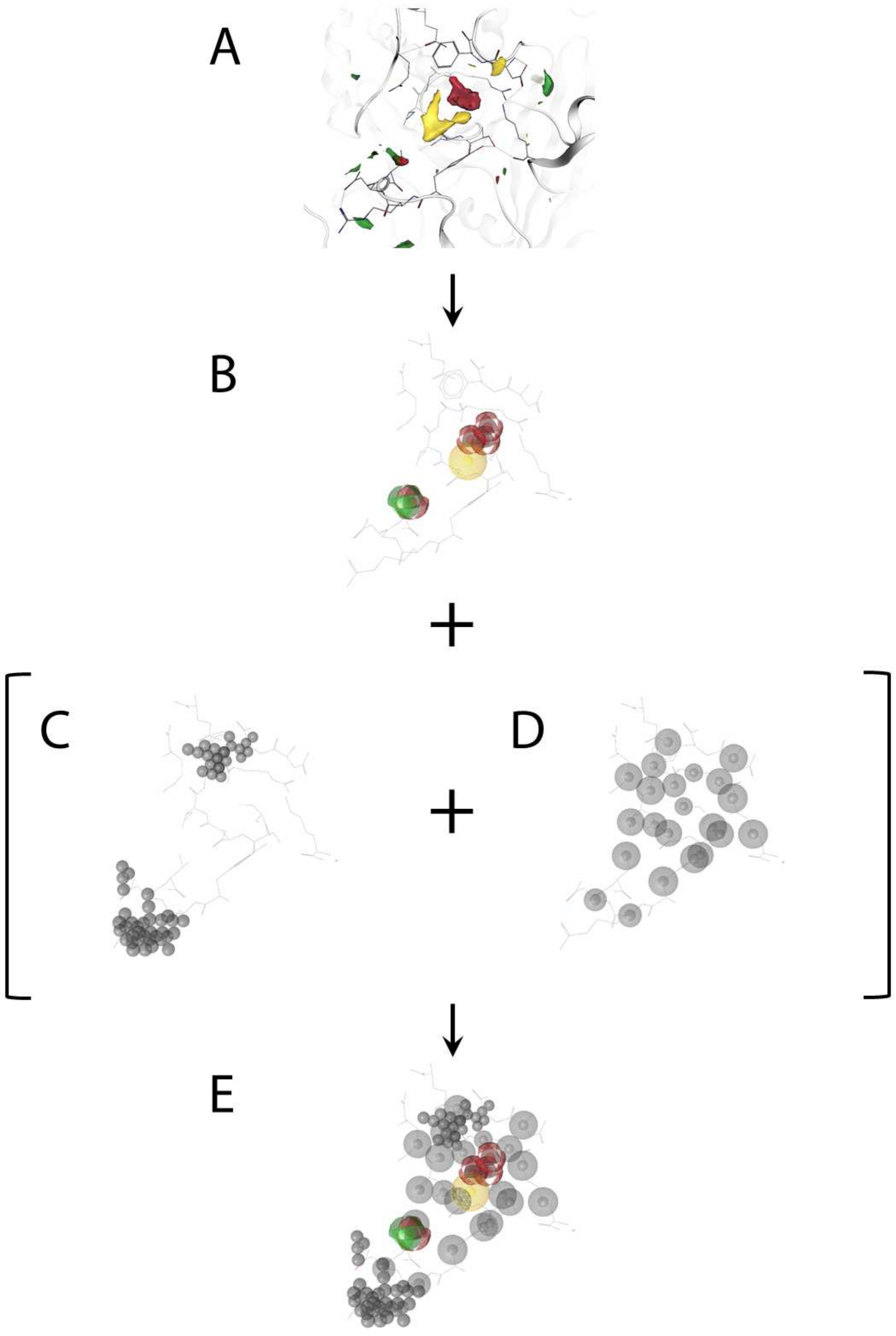


Figure 4.39.: Schematic representation of the pharmacophore generation workflow on the example of frame "1PTYorigFrame3290"; A: Molecular interaction field, B: pharmacophore translation of selected interaction energy maxima, C: automatically generated exclusion volumes based on selectivity pattern, D: manually added exclusion volumes, E: final interaction pattern; color code: green - hydrogen bond donor, MIF threshold -3.4; red - hydrogen bond acceptor, MIF threshold -3.4; yellow - hydrophobic, MIF threshold -1.7.

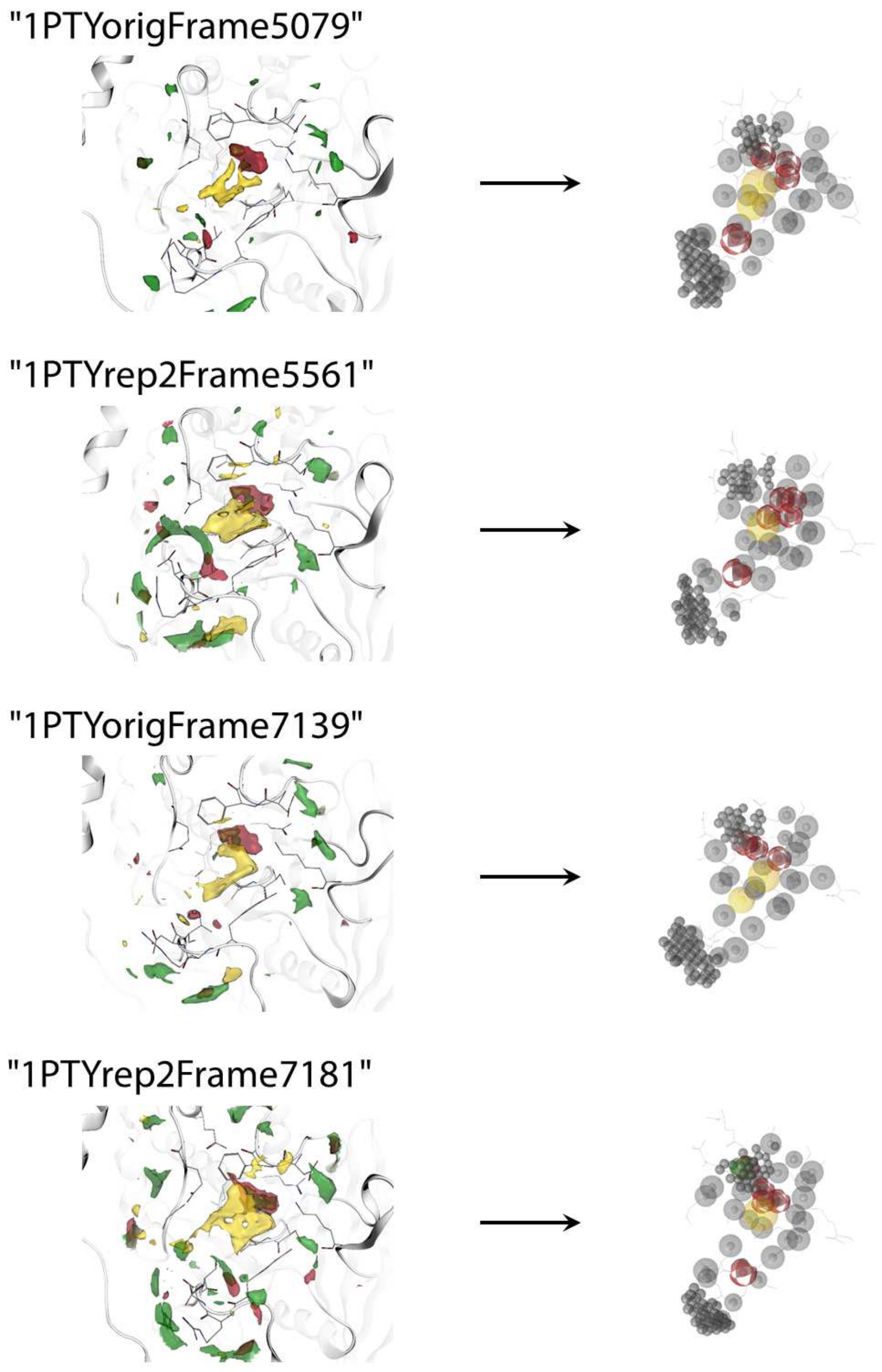


Figure 4.40.: Molecular interaction fields and final interaction patterns based on the selected frames "1PTYorigFrame5079", "1PTYrep2Frame5561", "1PTYorigFrame7139" and "1PTYrep2Frame7181"; color code: green - hydrogen bond donor, MIF threshold -3.4; red - hydrogen bond acceptor, MIF threshold -3.4; yellow - hydrophobic, MIF threshold -1.7.

In the next step, four databases of commercially available compounds were prepared as described in the Experimental Section and screened with the optimized selectivity and interaction patterns. The numbers of hits for each pharmacophore are summarized in Table 4.6.

Database	*mod_3290*	*mod_5079*	*mod_7139*	*mod_5561*	*mod_7181*
ChemBridge	3368	610	7356	1759	8
ChemDiv	2834	1107	12859	2352	7
keyorganics	599	197	1821	399	5
VitasM	4203	941	16176	2633	84

Table 4.6.: Virtual screening results (number of hits) of four vendor databases with the optimized interaction patterns.

Database	*mod_3290*	*mod_5079*	*mod_7139*	*mod_5561*	*mod_7181*
ChemBridge	326	58	403	108	0
ChemDiv	162	151	541	137	0
keyorganics	67	52	308	26	0
VitasM	359	113	1157	167	1

Table 4.7.: Results (number of hits) of the pharmacophore rescreening of the representative poses extracted from docking grouped by the four vendor databases.

All hits of the pharmacophore screening were submitted to protein-ligand docking into the corresponding frame to assess their most likely binding modes and in order to reassure in the next step the pharmacophore fit of the ligand in a protein bound conformation. 25 docking poses per ligand were generated and checked for consistency utilizing an R script created for that purpose (see Section A.4). The script performs a density-based clustering of the poses for each ligand and based on the number of clusters with sufficient size chooses up to three representatives. Figure 4.41 shows an example for the result of the density-based clustering for 25 docking poses of a small molecule in PTP1B. Two clusters are found beneath two single poses. For the identified clusters representative structures are selected. The results of the automatic processing nicely agree with intuitively chosen poses.

Hit numbers of the consequent pharmacophore rescreening of the representatives can be found in Table 4.7.

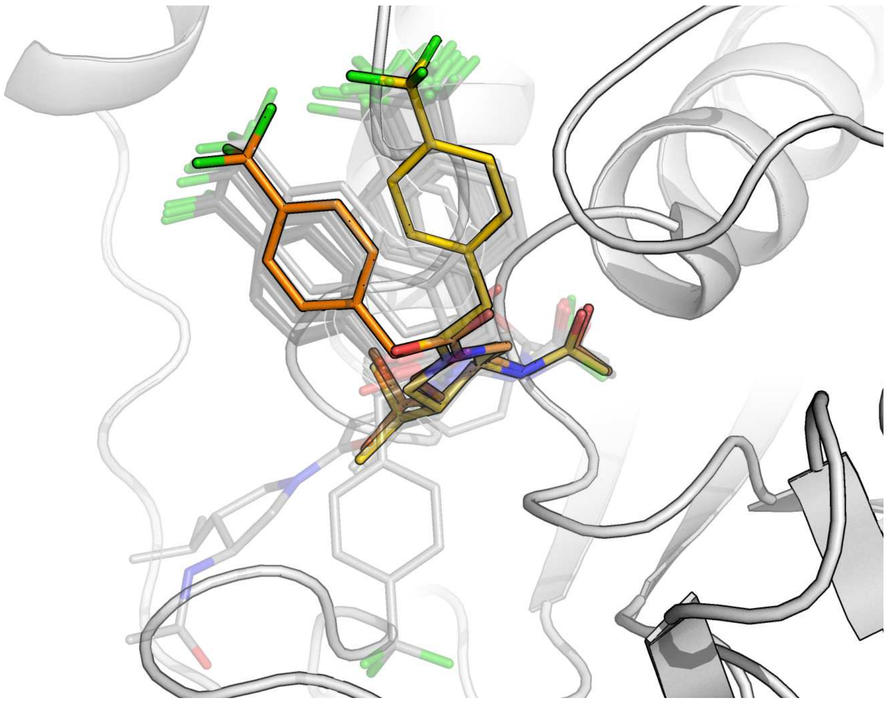


Figure 4.41.: Results of the density-based clustering in R for 25 docking poses of an exemplary small molecule in the PTP1B binding site; yellow and orange poses are chosen for further analysis, translucent poses are discarded.

The results show that the selection of consistent poses together with the pharmacophore rescreening leads to a significant reduction of hits. The number of hits was further reduced by reassuring that the selected binding poses additionally fit the selected protein frames well: Since the protein could show adaption to the bound ligand and move away from the conformational pattern correlated with selectivity, energy minimization of the poses selected via pharmacophore rescreening together with the surrounding amino acids was performed. This step was then followed by visual inspection of the energy minimized conformations with the previously described selectivity map. Only poses with protein conformations matching at least 100 of the 110 points were kept. After careful visual inspection considering both good protein-ligand complementarity as well as rigidity of the ligands to decrease the possibility of them fitting to other protein conformations and therefore to TC-PTP, the 14 compounds depicted in Figure 4.42 were chosen for biological testing.

The selected compounds show diverse rigid scaffolds different to those of known PTP1B inhibitors. Despite the different scaffolds, the compounds show some similarities. For example 9 of the 14 compounds show a trifluoro moiety and three of the 14 compounds are spiro-compounds. Since the ligands were already chosen to be as diverse as possible during visual inspection, the similarity of the resulting compounds suggests either a too restrictive filtering procedure or a lack of chemical diversity in the screened databases.

Figures 4.43 and 4.44 show docking poses of two of the selected compounds. Compound 0357-0002 shows good shape fit in the catalytic cavity as well as close to the YRD loop. It can interact with the protein via four hydrogen bond acceptor interactions and three charge interactions as well as a big area of hydrophobic interactions between the catalytic cavity and the YRD loop. The charge interaction to Arg47 has a high potential of exploiting a PTP1B exclusive conformation of the YRD loop. Additionally, there is a small empty place in the catalytic cavity close to Gln266 which could allow the accommodation of a water molecule as observed during catalysis and add water mediated interactions of the ligand to Gln266.

Compound 18735792 shows a similar good shape fit as compound 0357-0002. It is noteworthy that this compound is not negatively charged as most known PTP1B inhibitors, but in fact contains a tertiary amine which could positively influence the compounds cell permeability. Due to the absence of a negative charge, the interactions with the catalytic cavity are dominated by hydrogen bonds. The docking pose shows four hydrogen bonding interactions with the catalytic pocket, but they are assumed to be less strong then the ones detected for compound 0357-0002, due to the small size and therefor tightly bound lone-pairs of the fluorine atoms. As described for compound 0357-0002, additional water mediated hydrogen bonds of the ether oxygen to Gln266 seem possible. Additional hydrophobic interactions support the protein-ligand interactions in the active site. Further two more hydrogen bond acceptor and one hydrogen bond donor interaction are shaped to the YRD loop which increase the likelihood of the compound to prefer PTP1B selective conformations.

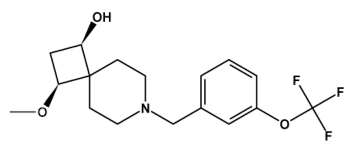
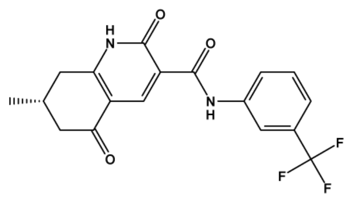
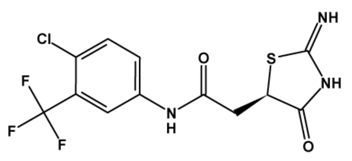
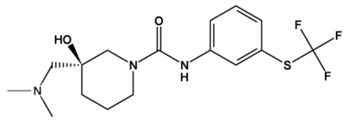
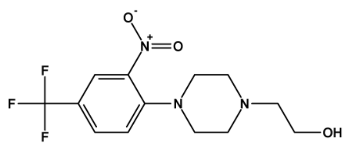
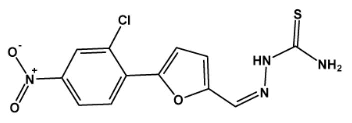
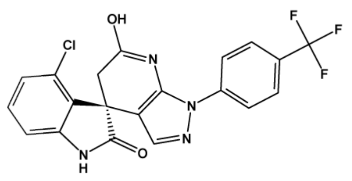
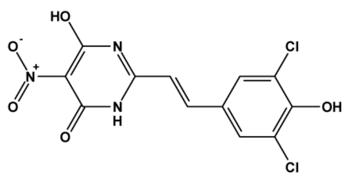
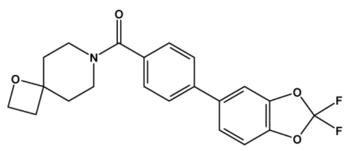
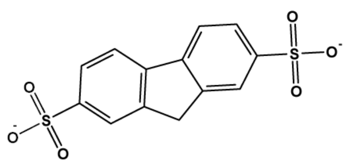
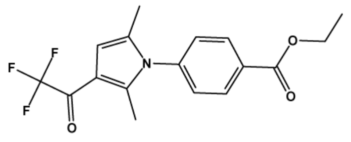
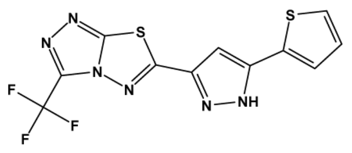
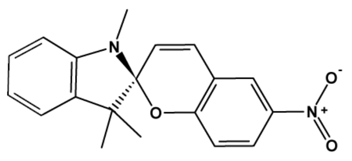
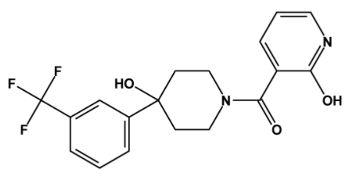
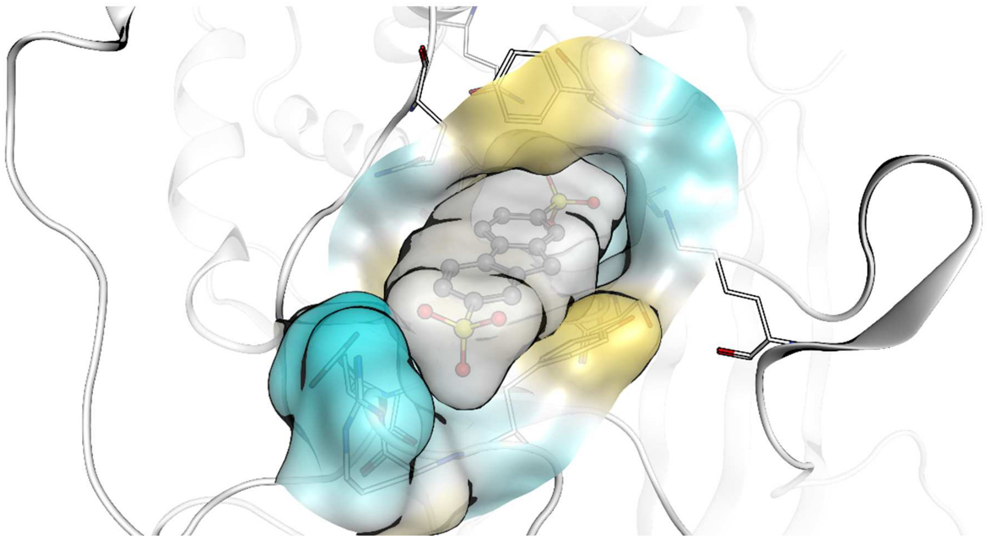
ID: 18735792  **Weight_neutr:** 345.3610	**ID:** STL198760  **Weight_neutr:** 364.3230	**ID:** 7213995  **Weight_neutr:** 351.7360
ID: 27986055  **Weight_neutr:** 377.4310	**ID:** 4265-2141  **Weight_neutr:** 319.2830	**ID:** STK361909  **Weight_neutr:** 324.7480
ID: STK655018  **Weight_neutr:** 432.7890	**ID:** STL370257  **Weight_neutr:** 344.1100	**ID:** 35547923  **Weight_neutr:** 387.3820
ID: 0357-0002  **Weight_neutr:** 324.3330	**ID:** MS-0941  **Weight_neutr:** 339.3130	**ID:** STL154686  **Weight_neutr:** 342.3290
ID: 0393-0130  **Weight_neutr:** 322.3640	**ID:** 58465174  **Weight_neutr:** 366.3390	

Figure 4.42.: Commercially available compounds selected for biochemical evaluation.



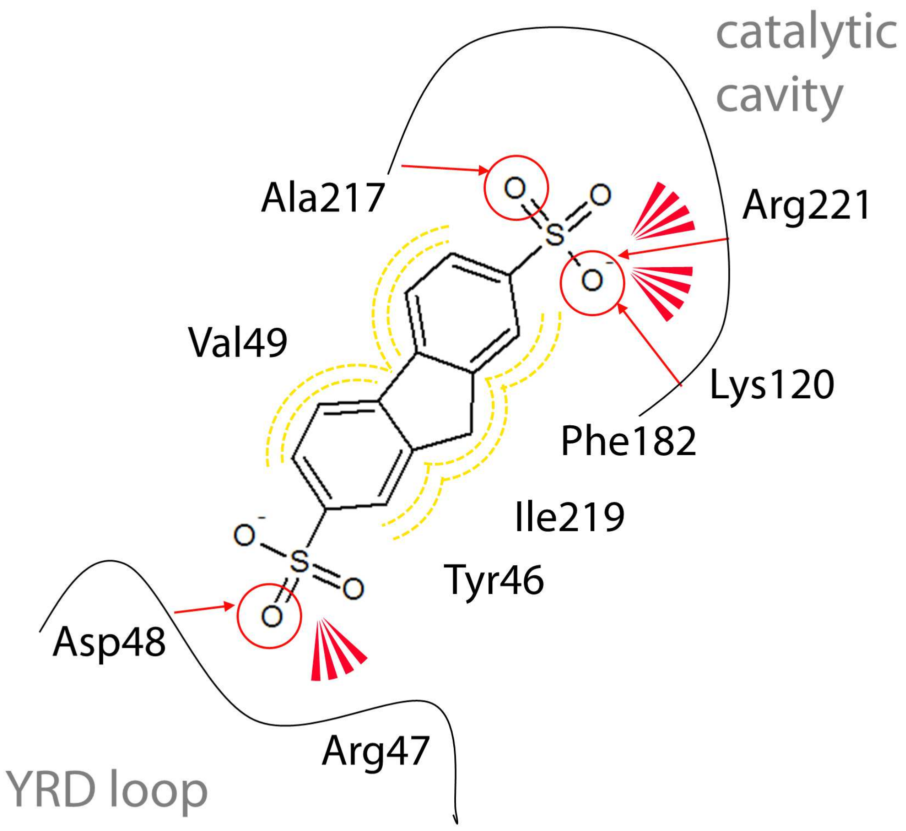
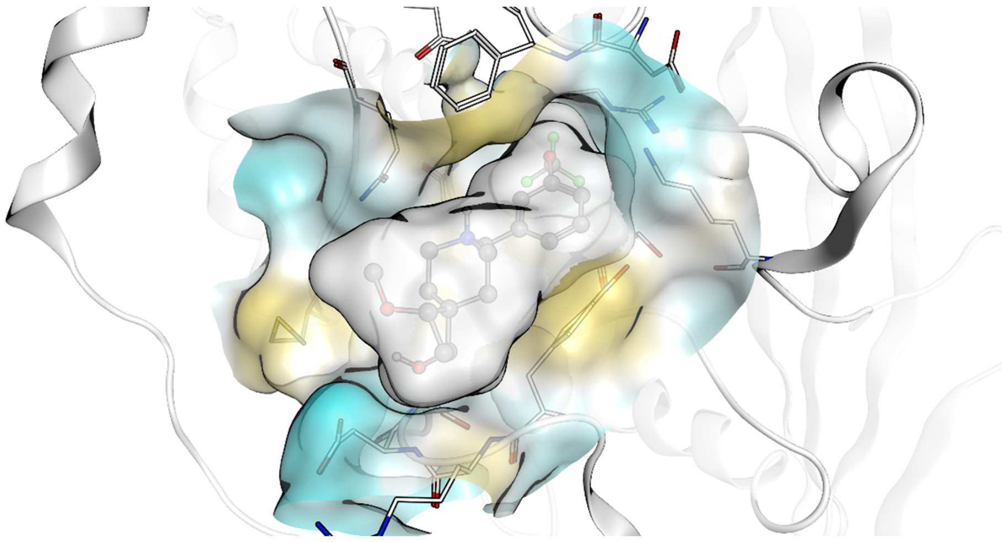


Figure 4.43.: Docking pose of compound 0357-0002 in the active site of PTP1B; Top: 3 dimensional depiction of protein binding site and ligand shape; Bottom: 2 dimensional ligand depiction with pharmacophoric interactions to the protein; red circles and lines: hydrogen bond acceptor, red star: negative ionizable, green: hydrogen bond donor, yellow: hydrophobic interactions.



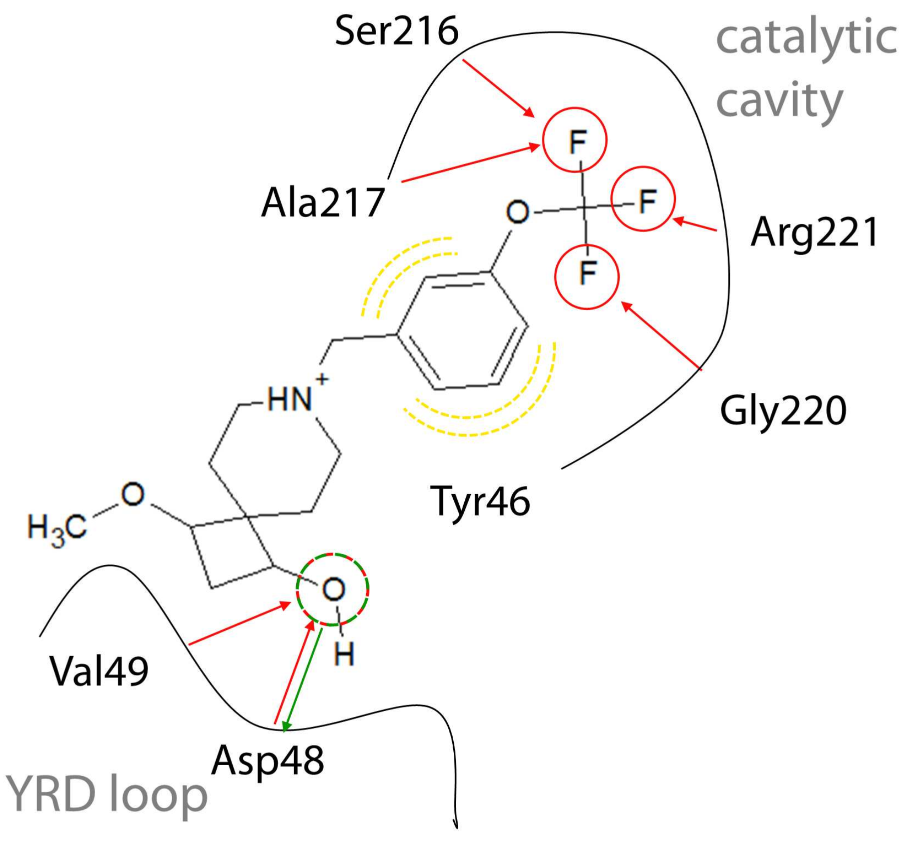


Figure 4.44.: Docking pose of compound 18735792 in the active site of PTP1B; Top: 3 dimensional depiction of protein binding site and ligand shape; Bottom: 2 dimensional ligand depiction with pharmacophoric interactions to the protein; red circles and lines: hydrogen bond acceptor, red star: negative ionizable, green: hydrogen bond donor, yellow: hydrophobic interactions.

5. Discussion

5.1. Part I: Protein-Ligand Interaction Analysis

The overall aim of this study was the detection of key features in the surrounding of the active site of PTP1B that could drive selectivity of PTP1B ligands. It was therefore assumed that despite the disappointing research experiences found in literature, the active site of PTP1B contains unique features that can be exploited to achieve selectivity of small molecule binders against TC-PTP.

The first section of the corresponding analysis deals with comparison of the protein sequences and of three-dimensional information from crystal structures. The high sequence similarity found in the active site area is not a new finding, still this detailed analysis lays important groundwork for further parts of this study. Additionally, the consideration of sequence differences together with the three-dimensional structure of PTP1B can explain, why the observed sequence differences in the active site are hard to target with ligand features. It is worth mentioning that the overall sequence identity of the two proteins is only 57% and therefore low in comparison to the active site similarity. This part of the study therefore supports the idea that differences in the overall protein structure could influence ligand binding and enable ligand selectivity. This idea was brought up many years before this study, however, so far no definite proof or reasonable mechanism was found for this hypothesis. Consequently, binding poses of co-crystallized selective PTP1B inhibitors were analyzed with different methods. Conventional methods like the comparison of protein-ligand interactions of selective and unselective PTP1B inhibitors show only subtle tendencies, but no solid explanation for the observed selectivities. Additional, the results suggest C site interactions, especially with Asp48, to influence selectivity against TC-PTP. The analysis of interaction counts also reveals the drawbacks of this method: Two crystal structures (1Q6T and 1Q6P) show crystal packing artifacts, which could introduce structural changes that distort the result. Further, while counting polar interactions like hydrogen bonds is rather intuitive, the quantification of hydrophobic interactions and the corresponding effects on entropy is challenging. Additionally, ligand binding affinities for different high affinity complexes have been related to shape complementarity or buried surface, a factor that is not accounted for by counting protein-ligand interactions. The question of PTP1B selectivity therefore seems unsolvable taking into account only conventional methods of analysis on static protein structures. Still conventional methods like analyzing protein-ligand interactions in static structures have been proven useful in many cases, therefore the absence of explanations in the case of PTP1B selectivity is an important result.

Since static protein structures did not reveal promising results, the flexible behavior of selective inhibitors in both proteins was then analyzed using molecular dynamics simulations. Contemplating the results of this part, however, it needs to be taken into account that the molecular dynamics simulations for TC-PTP were run on homology models using PTP1B template structures. Therefore it may simply not be possible for the TC-PTP structures to escape the possibly artificial local energy minimum created by homology modeling.

RMSD and RMSF plots of the conducted simulations did not reveal meaningful differences in protein flexibility between the two proteins. However, they all support that the region of the pTYR loop and the connected α-helix is the most stable part of the protein. Therefore, for further analyses the molecular dynamics frames were aligned on this stable region to get a picture about flexibility of the surroundings given a ligand anchored in the catalytic cavity. Remarkably, the WPD loop known for its high flexibility shows very low flexibility over the time of the simulations, probably due to fixation in the closed state by the complexed ligands.

To investigate protein-ligand interactions over the molecular dynamics simulations, dynophores were used. Similar to the static interaction analysis the variation of the results was high and allowed no clear deductions regarding differences of both proteins. However, according to the dynophore analyses, B site interactions seem more stable for PTP1B which is in agreement with their relevance for selectivity deduced form the static interaction counts and also found in literature. Unfortunately, for those interactions the variations between simulations of one protein are especially high, which makes this result highly unreliable. However, this analysis also suggests parts of the A site and the C site as possible interaction sites to introduce selectivity. In the simulations of PTP1B-complex 1XBO and the corresponding homology model a slight shift of the ligand between A and B site in TC-PTP compared to PTP1B could be observed which seems to be connected to different hydrophobic interactions in the region connecting both sites. Unfortunately, the dynophore results share the drawback of static interaction counts that hydrophobic interactions are hard to quantify and shape complementarity is not sufficiently accounted for.

Consequently, we developed a new method to assess protein-ligand shape complementarity on frames of a molecular dynamics trajectory. Surprisingly, the results on the test case 1XBO indicate that flexible shape complementarity might be the previously undiscovered key feature of PTP1B selectivity. Interestingly, this method suggests the A site interactions to play a main role in the selectivity of the 1XBO ligand, which agrees with the result of the dynophore analysis that suggests HBA3 to be selectivity relevant. Additionally, the detected difference in HYD1 interactions from the dynophore analysis matches the only detected atom outside the A site, C35. Further, the strain energy plot did not indicate significant differences in ligand conformations. It stands out that the atoms with highest differences in ligand RMSF values match the atoms with shape fit differences. The increased activity of the ligand in PTP1B could therefore be explained by tighter binding due to better shape complementarity of the protein to the ligand in the catalytic cavity. Also for the second test case, 1QXK, the new method indicated that shape fit and here additionally ligand strain could be responsible for the observed selectivity of the ligand. Surprisingly, for the 1QXK ligand, despite an overall better shape fit to PTP1B, the active site part of the ligand shows increased shape fit to TC-PTP, which fits the tendency observed in the dynophore statistics where HBA3, HBA4 and HBD1 show higher mean occurences in TC-PTP. Here, B and C site interactions seem responsible for the selectivity and connected to especially good shape fit. As for the B site, this good shape fit is correlated with reduced flexibility of the ligand in this area like observed for 1XBO. For the C site, however, the increased shape fit does not correlate with a difference in ligand flexibility. One possible explanation for that could be the association of good shape fit in this area with the high number of frames with low ligand strain in PTP1B: a PTP1B exclusive protein conformation of the YRD-loop could enable ligand relaxation compared to other protein conformations which possibly introduce ligand strain in order to enable hydrogen bonding at N1(HBD4) and N2(HBD3). This hypothesis is supported by the result of the dynophore analysis indicating higher occurences of the relevant hydrogen bonds in PTP1B as well as the results of Binding Site Shape Clustering section.

Since the novel method of protein-ligand shape complementarity yielded interesting results for the PTP1B/TC-PTP testcases in this study, investigations on additional testcases would be highly interesting to discover the applicability of the protein-ligand shape complementarity method for other other projects and proteins. Unfortunately, since this method is in the current implementation highly time and resources consuming, we had to refrain from investigating further test cases. In order to assess its relevance on further test cases reimplementation therefore might be a rational step. Additionally, supplementary studies on the impact of results on input settings of the method like the size and shape of the starting point map and the spacing of the grid points would be desirable to optimize the method for general use. Overall, this method represents a novel opportunity to look at protein-ligand interactions from a different viewpoint and could help to explain ligand activity differences in so far undisclosed cases.

5.2. Part II: Binding Site Shape Clustering and Screening

The second part of this study deals with the question if the differences discovered in protein-ligand complementarity can be targeted to find or create inhibitors of increased selectivity. Hence, it is assumed that the differences in shape complementarity originate in differing conformational preferences of the two proteins.

Consequently, molecular dynamics simulations of protein structures for both PTP1B and TC-PTP with only a ligand anchor to keep the WPD loop closed were performed and analyzed. Keeping the WPD loop closed was considered appropiate due to most higher active inhibitor binding poses showing a closed WPD loop. Furthermore, this approach aims to separate the movements connected to the WPD loop opening and closure from movements relevant for selectivity of active site inhibitors. However, there is no guarantee that this restraint on the WPD loop is not suppressing relevant movements or introducing unnatural behavior, especially in TC-PTP, into which the anchor ligand was just placed instead of using protein-ligand docking. Additionally, the selection of the starting structure of PTP1B and the created homology model for TC-PTP could impact the outcome of the analysis, as could parameters like size and position of the starting point map as well as grid spacing or the choice of the clustering algorithm. Especially, due to the insufficient placement of grid points on the WPD loop, loop movements could not be properly detected in this analysis. However, due to the anchor ligand they were assumed to be negligible. Again, investigation of all influencing factors was out of the scope of this thesis and leaves room for further investigations of this novel method.

Overall, the method yields interesting results with rationally guided selection of input parameters, showing differing conformational preferences for the two proteins:

Firstly, clustering of the three bootstrapping samples leads to a good agreement of the single results regarding number and size of the detected clusters with nice separation between PTP1B and TC-PTP clusters. Secondly, the grid points selected to be selectivity relevant by the method show a high amount of overlap between the results of the bootstrapping samples.

Translation of this information into a selectivity point map allows visualization of the result and exploitation for filtering of molecular dynamics frames. This way it could be verified that filtering with the map selects PTP1B frames (905 frames selected), but not a single TC-PTP frame. Additionally, PTP1B frames matching the selectivity map could be chosen as representative frames for the consequent virtual screening workflow.

Interestingly, the points detected as relevant for selectivity concentrate around only two protein sites: the first site is located around the side chain of Phe182. The detected differences here seem to correlate only with rotations of the phenyl ring of this amino acid. A detection of an area in the A site with this method corroborates the results of the first part of this study, which indicates this site as selectivity relevant for some of the analyzed ligands. However, these differences could be induced by the artificial placement of the phenylphosphate moiety in the active site for loop closure during homology modeling. The second part of the selectivity map is located around Arg47 and Asp48 in the C site of the binding area. Here the YRD loop seems more flexible in PTP1B and therefore able to adapt a conformation more bent towards the catalytic cavity. Astonishingly, also the static protein-ligand interaction analysis as well as the dynophore analysis of ligand 1QXK pointed to this site to be relevant for selectivity. Surprisingly, no points in the B site were detected to be relevant for selectivity. This could correlate with the high intra-protein variance discovered in the dynophore analysis. Additionally, the alignment on stable A site atoms could lead to higher variation in more distant points. Both factors could disturb a pattern of occupancy differences and render it undiscoverable by the clustering method.

The final part of this study aims at exploiting conformational differences of PTP1B and TC-PTP to find selective PTP1B inhibitors. Of the 905 frames matching the selectivity map, 5 were chosen for the virtual screening workflow. The number was limited by the available computing time for screening, therefore diverse frames were picked from the set. Possibly more elaborate picking methods like choosing frames with many close neighbors could increase the chance to find relevant protein conformations.

In the attempt to not only include spatial, but also pharmacophoric information to the screening to increase the likelihood of activity at the protein, molecular interaction fields were used to create interaction patterns screenable with LigandScout. Different to classical 3D pharmacophores, these interaction patterns are not based on one or more ligands. This has advantages and disadvantages: On the one hand, purely protein structure based interaction patterns allow inclusion of interactions with high likelihood of positive impact on activity independent on whether or not they are already present in known ligands. On the other hand, a good interaction energy score at a certain grid point does not necessarily imply relevance of this interaction for activity. In fact, the good score might be related to the energy function or the interaction point could be too small or restricted in its angle to be met by a real ligand. However, this protein structure based selection of interaction points is a common approach, which was implemented in a similar fashion in Discovery Studio/Catalyst and the most recent LigandScout version [113, 80, 81].

The results of this method are corroborated by the high resemblance of the interaction patterns to 3D pharmacophores of known PTP1B ligands with at least three nested hydrogen bond acceptor features in the catalytic cavity and a close-by hydrophobic feature below the WPD loop. Due to the additional C site interactions optimized to fit only selective PTP1B conformations, there is no appropriate data set for validation of the interaction patterns. Validating the interaction patterns on PTP1B inhibitors by omitting their C site features would also be of limited relevance, since the remaining interaction patterns would only contain a few number of features and most likely be very unspecific. Their high similarity to pharmacophores of known ligands in the active site therefore depicts the only validation method.

The applied combination of screening with the chosen interaction patterns, protein-ligand docking, pose-consistency filtering and interaction pattern rescreening was chosen to reduce false positives among the screening hits. As mentioned in the "Computational Methods" section combinations of filtering and or screening methods are not uncommon to decrease false positive rates is virtual screening. However, most studies rely on docking scores rather than pose-consistency filtering. Pose-consistency filtering was developed and used in this study, since docking scores were found to be unreliable in many cases, especially if they were not adapted and validated on the relevant target. Although pose-consistency is depending on the created docking poses and therefore also on the scoring function, it gives a more qualitative decision criterion for pose selection which is not dependent on the total score and factors like ligand size.

The combination of conformational rigidity, good shape fit of the ligand to the selective protein frame and matching of the interaction patterns depicts a very restrictive filter for small molecules. Therefore it seems reasonable that the resulting compounds from vendor databases show a lack of diversity and partially unsatisfactory agreement with the requirements like the trifluoro-group as triple hydrogen bond acceptor features. Since all of the mentioned aspects ensure integral properties of the desired ligands, the only option to find more customized ligands would be *de novo* synthesis. However, this was out of the scope of this study, but would be a desirable step after an initial biochemical validation of the workflow. Further, fluorinated motifs have been discovered to adopt diverse protein binding features and could therefore be an interesting alternative to known phosphotyrosine mimetics [114].

In total the second part supports the hypothesis discovered in the first part of the study, suggesting that differences in conformational flexibility of both proteins exist. In fact, the results even suggest a possibility to exploit those differences by small catalytic site binding compounds.

6. Conclusion and Outlook

Despite decades of research, the key principles of PTP1B selectivity against TC-PTP are still unknown and highly selective PTP1B inhibitors have not been published. Since selective inhibition of PTP1B could be a valuable contribution in the fight against deseases like Type 2 diabetes and obesity, this study strived to detect and exploit key features of PTP1B selectivity. This aim was persued in two different approaches: a detailed analysis of partially selective PTP1B inhibitors in their protein bound conformations in Part I and a consequent predictive method based on clustering of binding site shapes derived from molecular dynamics simulations in Part II.

Since classical interaction analysis approaches did not lead to concise results regarding influencing factors of PTP1B selectivity, a novel method to measure flexible protein-ligand shape complementarity was developed and implemented. Validation on two partially selective ligands led to interesting findings, indicating protein-ligand shape complementarity and resulting increased ligand stability in different areas of the binding site to be responsible for the selectivity of the investigated ligands. Additionally, for one of the ligands the results point towards an increased possibility of PTP1B to adapt to the ligand with the YRD loop.

Those conclusions were corroborated by the second part of the study: The novel method of binding site shape clustering developed here also implicated a part of the catalytic cavity as an area of differing protein flexibility in PTP1B and TC-PTP and therefore relevant for selectivity. However, due to the limited space in the catalytic cavity and the highly similar interaction feature patterns in both proteins, the optimization of ligands towards a PTP1B preferred conformation at this site would be extremely difficult. Interestingly, this method further strongly suggested the C-site of the protein binding site to show different conformational preferences in both proteins. Opposed to the differences in the catalytic cavity, these differences show increased potential for exploitation with small ligands. Consequently, the method was extended to exploit the detected selectivity shape features for virtual screening. With this approach, commercially available compounds with high potential to be selective PTP1B inhibitors could be identified.

Beneath the interesting results on the test case of PTP1B, the two novel methods developed in this study represent valuable new tools that could be of use for different applications: The shape complementarity tool complements the shortcomings of classical protein-ligand interaction analysis tools by focusing on ligand conformational preferences and shape fit, which have been shown to be integral parts of ligand affinity and could therefore help to explain and exploit activity differences especially in similar protein-ligand complexes. The binding site shape clustering tool addresses the issue of different conformational preferences of similar proteins for exploitation in drug development projects. It could therefore guide development of selective ligands even in cases where interaction patterns of the proteins are highly similar. However, both methods are only valuable in combination with interaction feature based methods like pharmacophores or molecular interaction fields to ensure an overall sufficient complementarity of a ligand to the target.

Overall, the results of both parts of this study concur nicely in indicating shape complementarity as an important feature to increase PTP1B selectivity. Additionally, the novel Binding Site Shape Clustering method developed in this study made it possible to exploit this discovery for virtual screening and both aspects together could positively impact future drug design projects, where selectivity plays an important role.

7. Experimental Section

Since this thesis focuses on computational methods, this part primarily contains details and settings of the conducted computational steps that would disturb the explanations in the "Computational Methods" section.

7.1. Part I: Protein-Ligand Interaction Analysis

7.1.1. Sequence Comparison of PTP1B and TC-PTP

Protein sequences for PTP1B (P18031) and Isoform 1 of TC-PTP (P17706-1) were extracted from Uniprot [110] and aligned with default settings for proteins using Clustal Omega [115]. PDB [46] complex 1PTY [116] was chosen as representative for analysis of binding site residues.

7.1.2. PTP1B Crystal Structure Analysis

The Uniprot referenced PDB structures for PTP1B were sorted by resolution and complexes with worse or equal resolution than 2.70 Å were excluded. Structures with covalent active site or nearby (C_α in 15 Å sphere around GLN262-CD) modifications like mutations, intermediates or missing loops were excluded, since they could disrupt the natural protein conformation and due to the high similarity of the two proteins subtle changes could distort features of selectivity. Also excluded were structures with residue modifications connected to modified activity (see e.g. [117] or [118]) and structures with no ligand or only allosteric ligand. For the selected structures, the Chembl database [107] and, if not available there, the original publication were checked for ligand activities of the co-crystallized ligand against PTP1B and TC-PTP. If available, activities of the ligand against both proteins from the same group and assay and under comparable conditions were preferred. Selectivity factors were calculated and the 10 most selective structures were chosen additionally to 4 non-selective complexes. Analysis of the selected complexes was performed using MOE(2015.10) [43] and LigandScout(4) [80, 81]. Interaction frequencies were plotted with R [89].

7.1.3. TC-PTP Crystal Structure Analysis and Homology Modeling

Backbone alignment for visual inspection of PTP1B structure 1PTY and TC-PTP structure 1L8K was performed in MOE(2015.10) [43]. Homology modeling was also performed using MOE(2015.10) [43]: The TC-PTP sequence was aligned to the sequence of the PTP1B template being one of the crystal structure complexes 1QXK [119], 1Q1M [120] and 1XBO [121]. C- and N-terminal outgap modeling was disabled, automatic disulfide bond detection was enabled and atoms of the co-crystallized ligand were selected as environment for induced fit. The basic modeling step was chosen to create 25 main chain models with 10 side chain samples each at 300 K. Amber99 was chosen as forcefield . Model refinement was performed in mode "Fine" with RMS Gradient set to 1.0. The resulting 250 models were analyzed regarding their protein and ligand RMSD to the modeling template.

7.1.4. Molecular Dynamics Simulations

Basic protein preparation was performed in MOE(2015.10)[43]: sidechain protonation was performed with the Protonate 3D tool at pH=7. The cysteine residue in the catalytic center was deprotonated manually. All molecular dynamics simulations were performed with the software Desmond 3.1 [71]. Systems were set up in an orthorhombic box with SPC water and neutralized with Na^+ ions. NaCl was added to a concentration of 0.15 M. Simulations were performed with NPT ensemble at 300 K and 1.01325 bar under periodic boundary conditions. A short relaxation time was followed by a 25 ns production run. For relaxation the Desmond standard NPT relaxation protocol was used. Energies were recorded every 1.2 ps and the trajectory was saved every 4.8 ps. As forcefield OPLS2005 was chosen. Calculations were run in triplicates on 24 CPUs each on the Soroban computing cluster of FU Berlin. Trajectories were analyzed using VMD [122] and the R package "bio3d" [89, 123].

7.1.5. Dynophore Analysis

The dynophore analysis is based on the dynophore tool for statistical analysis of LigandScout 3D pharmacophores, which was recently developed in our group by Gerhard

Wolber and Dominique Sydow [34, 35, 80, 81]. Additional statistical analyses of the results were carried out in R [89].

The dynophore application creates a 3D pharmacophore for each frame of a molecular dynamics trajectory. Pharmacophore features with the same feature type and interacting atoms on the ligand are binned into superfeatures. Superfeatures are then statistically analyzed for example regarding its occurrence over the molecular dynamics simulation and the distances of the interacting atoms on protein and ligand side.

7.1.6. Shape Complementarity Analysis

The modification of the POVME tool [94, 95], which was used in this study can be found in section A.1 of the Appendix. For generation of the POVME point maps grid spacing of 0.5 Å was used. The distance cutoff for ligand heavy atoms to points of the created maps was set to 0.59 Å for the inner map and 1.59 Å for the outer map. The distance cutoff for protein heavy atoms to the created point maps was kept at 0.59 Å.

7.2. Part II: Binding Site Shape Clustering and Screening

7.2.1. Generation of Input Data

Homology modeling was performed using MOE(2015.10) [43]: The TC-PTP sequence was aligned to the sequence of the PTP1B template 1PTY [116]. The ligand was reduced to its phenylphosphate moiety and not used for adaptive homology modeling. The basic modeling step was chosen to create 25 main chain models with 10 side chain samples each at 300 K. As forcefield Amber99 was chosen. Model refinement was performed in mode "Fine" with RMS Gradient set to 1.0. The resulting 250 models were analyzed and filtered regarding the similarity to the PTP1B template in the binding area. After selection of the final model, the phenylphosphate moiety and the adjacent catalytic water molecule were transposed to the TC-PTP homology model.

Molecular dynamics simulations were prepared and conducted as described in the subsection Molecular Dynamics in Part I of the methods section. Simulations were run in triplicates over 50 ns each after a short relaxation run.

For the creation of point maps using the modified POVME script (section A.1 of the Appendix) a starting point map of 10427 points spanning the whole binding area was created. Grid spacing was set to 1.0 Å and the distance cutoff of protein atoms to points of the created maps was fixed to 1.09 Å. A contiguous pocket seed sphere was placed in the center of the catalytic cavity.

7.2.2. Selectivity and Interaction Patterns

Molecular interaction fields were created using MOE2015.10 [43]. Three probes were selected for interaction field generation: "N1" (amide NH group) for hydrogen bond donor interactions, "N:" (nitrogen atom with lone pair) for hydrogen bond acceptor interactions, "DRY" for hydrophobic interactions.

7.2.3. Virtual Screening

For virtual screening databases of the vendors ChemBridge, ChemDiv, VitasM and key-Organics were chosen. Vendor databases were downloaded in the version of July 2016. The databases were prepared removing salt and solvent molecules from compound entries. Then standardization of the structures was performed using the group protocol for the ChemAxon Standardizer software [45]: amidine and guanidine structures were harmonized, different acidic structures were deprotonated and basic structures were protonated. 25 conformations per ligand were generated with LigandScout 4 [80, 81]. After the preparation the databases contained the following numbers of molecules: ChemBridge - 1,141,083; ChemDiv - 1,455,331; VitasM - 1,384,271; keyOrganics - 79,617.

Screening of the databases and rescreening of selected docking poses with the molecular interaction field derived interaction patterns was conducted with LigandScout 4 [80, 81]. Protein-ligand docking of the resulting hits into the corresponding molecular dynamics frames was performed with GOLD version 5.2.2 [54]. The phenylphosphate moiety and all solvent molecules were removed prior to docking. The binding site was restricted to a sphere of 12 Å radius around the sulfur atom of CYS215. 25 poses per ligand were generated. Scoring of the docking poses was performed using the scoring function PLP with rescoring using Goldscore.

Density based clustering on the RMSD matrix of each molecule's 25 docking poses was performed using the package "dbscan" in R with the options eps = 1.5 and minPts = 8 [89].

8. Summary

Protein Tyrosine Phosphatase 1B (PTP1B) is a validated drug target for the treatment of diabetes type 2 and obesity. Until now, development of suitable modulators has been hampered by the polarity of the binding site and related bioavailability issues of the molecules. The design of selective inhibitors of PTP1B against the closely related T-Cell protein tyrosine phosphatase (TC-PTP), which was associated to severe side effects in animal studies, proved even more challenging. Over the years progress was made, but known PTP1B inhibitors only achieved at maximum moderate selectivity over TC-PTP.

This study aims to break the traditional boundaries of PTP1B selectivity by deliberately exploiting structural differences of both proteins. Due to their high similarity this requires thorough analysis of their static structures as well as their flexible behavior. The goal was therefore pursued with two different approaches:

In Part I of the study a detailed analysis of protein complexes with selective ligands was performed including their flexible behavior as determined from molecular dynamics simulations. Since common analysis methods were not able to explain the selectivity of the investigated ligands, a new method was developed which is able to assess parameters of ligand affinity that are not covered by currently available methods: steric complementarity of the ligand to the protein together with ligand strain. The developed tool allows to assess those properties on high numbers of molecular dynamics frames to calculate ligand shape fit in a flexible context. It further enables to trace back the ligand atoms or parts responsible for good or bad shape fit.

In Part II of this study the flexible behavior of the apoproteins was studied. Since surface properties are highly similar in both proteins, the analysis focused on binding site shapes. For this, a novel approach was chosen that translates binding site shapes from molecular dynamics simulations into point maps and subsequently uses clustering and difference calculations to find a PTP1B conformation most unlike to all discovered TC-PTP conformations. The workflow includes calculation of a selectivity map consisting of points in the binding site where highest and most relevant differences in PTP1B

and TC-PTP conformations occur. This map can be visualized and used for screening of selective PTP1B frames, which can then be used for protein structure based virtual screening for potentially selective PTP1B inhibitors.

Results of Part I indicate shape fit as the previously undiscovered reason for selectivity of some known PTP1B inhibitors. They further suggest that selectivity can be achieved by interactions in the catalytic cavity as well as in previously suggested areas (the B and C site) of the binding site. Additionally, the discovered reduced ligand strain in PTP1B for one of the analyzed ligands, while almost maintaining same occurences of protein-ligand interaction features, lead to the assumption that PTP1B possesses a higher ability to conformationally adapt to the ligand than TC-PTP. This assumption is corroborated by the results of Part II: The selectivity map indicates the catalytic cavity and the YRD-loop (C site) as areas of different flexible behavior. Especially the YRD-loop shows increased flexibility in PTP1B compared to TC-PTP. Additionally, ligands that have a high likelihood of exploiting the discovered conformational differences while still showing sufficient activity in PTP1B could be found in databases of commercially available molecules.

Overall, the two innovative approaches to discover key factors of PTP1B selectivity did not only lead to interesting findings, but could also be adapted to promote other drug design projects where selectivity is crucial but hard to achieve.

9. Zusammenfassung

Das Enzym Protein Tyrosin Phosphatase 1B (PTP1B) ist ein validiertes Wirkstoffziel für die Behandlung von Diabetes Typ 2 und Übergewicht. Die Entwicklung passender Modulatoren wurde immer wieder durch die Polarität der Bindetasche und damit verbundene Bioverfügbarkeitsprobleme der Moleküle zurückgeworfen. Noch schwieriger ist es aber, die Moleküle so zu modifizieren, dass sie Selektivität gegenüber dem nahe verwandten Enzym T-Zell Protein Tyrosin Phosphatase (TC-PTP) erhalten, was aufgrund der mit diesem Protein assoziierten starken unerwünschten Wirkungen notwendig erscheint. Trotz einiger Fortschritte in den letzten Jahren erreichen bekannte PTP1B-Inhibitoren bis jetzt maximal moderate Selektivität gegenüber TC-PTP.

Diese Arbeit hat es sich zum Ziel gemacht die durch bisherige Forschung gesetzten Grenzen der PTP1B-Selektivität zu durchbrechen, indem systematisch strukturelle Unterschiede der beiden Proteine ausgenutzt werden. Aufgrund ihrer sehr großen Ähnlichkeit erfordert dies eine genaue Analyse der statischen Proteinstrukturen sowie ihres dynamischen Verhaltens. Um dieses Ziel zu erreichen, wurden zwei Ansätze gewählt:

In Teil I der Arbeit wurde eine detaillierte Analyse von Protein-Komplexen mit selektiven Liganden und deren dynamischen Verhaltens mithilfe von Moleküldynamiksimulationen durchgeführt. Da verfügbare Analysemethoden nicht in der Lage waren die beobachtete Selektivität zu erklären, wurde eine neue Methode entwickelt, welche es ermöglicht zusätzliche Faktoren für Ligandenaffinität zu erfassen: sterische Komplementarität und konformationelle Energie des Liganden. Die entwickelte Methode ermöglicht es diese Faktoren an einer großen Anzahl von Moleküldynamik-Schritten zu erfassen, um Verteilungen für die sterische Komplementarität am flexiblen Protein-Liganden-Komplex zu erhalten. Zusätzlich können die Anteile eines jeden Ligandenatoms an der Gesamtkomplementarität berechnet und dargestellt werden.

In Teil II der Arbeit wurde das flexible Verhalten der Apoproteine näher untersucht. Da die Oberflächeneigenschaften beider Proteine kaum Unterschiede zeigen, konzentrierte sich diese Analyse auf die Form der Bindetaschen. Dafür wurde ein neuartiger

Ansatz gewählt, der die Form der Bindetasche aus Moleküldynamiksimulationen extrahiert und in Punktwolken übersetzt. Diese werden anschließend geclustert, woraufhin mithilfe von Distanzberechnungen PTP1B-Konformationen ermittelt werden, die den größtmöglichen Unterschied zu allen ermittelten TC-PTP-Konformationen aufweisen. Der dafür verwendete Workflow berechnet zusätzlich einen Selektivitätsfilter bestehend aus denjenigen Punkten der Punktwolke, die die größten beziehungsweise relevantesten Unterschiede in den PTP1B- und TC-PTP-Konformationen zeigen. Dieser Filter dient einerseits dazu die Selektivitätsrelevanten Bindetaschenareale zu visualisieren, kann aber auch zum Filtern nach selektiven PTP1B-Konformationen aus Moleküldynamiksimulationen dienen, welche anschließend zum strukturbasierten virtuellen Screening nach potentiell selektiven PTP1B-Inhibitoren genutzt werden können.

Die Ergebnisse von Teil I deuten darauf hin, dass die sterische Komplementarität den bisher unbekannten Grund für die Selektivität einiger bekannter PTP1B-Inhibitoren darstellen könnte. Außerdem geht aus den Analysen hervor, dass Selektivität womöglich auch durch Interaktionen mit der katalytischen Tasche hervorgerufen werden kann, neben Interaktionen mit schon in früheren Studien vorgeschlagenen Arealen der Bindetasche (B- und C-Seite). Zusätzlich zeigt einer der selektiven Liganden eine teilweise reduzierte konformationelle innere Enenergie, obwohl kaum Unterschiede im Vorkommen der Häufigkeiten der Protein-Liganden-Bindungen zu erkennen sind. Hier kann die Vermutung aufgestellt werden, dass dies durch eine erhöhte Fähigkeit von PTP1B im Vergleich zu TC-PTP verursacht wird seine Konformation an den Liganden anzupassen. Diese Vermutung wird durch die Ergebnisse aus Teil II der Arbeit gestützt: Der Selektivitätsfilter weist einerseits auf die katalytische Bindetasche, andererseits auf den YRD-loop (C-Seite) als Areale unterschiedlicher Flexibilität hin. Insbesondere der YRD-loop zeigt erhöhte Flexibilität in PTP1B im Vergleich zu TC-PTP. Zusätzlich konnten in diesem Teil der Arbeit Liganden in Datenbanken käuflich erwerbbarer Moleküle gefunden werden, welche nach unserem Modell eine große Wahrscheinlichkeit haben die ermittelten konformationellen Unterschiede bei zusätzlich guter Aktivität in PTP1B gezielt auszunutzen.

Insgesamt führten die zwei innovativen Ansätze zur Ermittlung von Schlüsselfaktoren der PTP1B-Selektivität nicht nur zu äußerst interessanten Ergebnissen, sondern könnten auch angepasst werden um Wirkstoffdesignprojekte voranzutreiben, bei denen Selektivität von großer Wichtigkeit, aber schwer zu erreichen ist.

Bibliography

[1] Z. Y. Zhang. Protein tyrosine phosphatases: structure and function, substrate specificity, and inhibitor development. *Annu Rev Pharmacol Toxicol*, 42:209–34, 2002.

[2] D. Barford, A. K. Das, and M. P. Egloff. The structure and mechanism of protein phosphatases: insights into catalysis and regulation. *Annu Rev Biophys Biomol Struct*, 27:133–64, 1998.

[3] N. K. Tonks. Protein tyrosine phosphatases - from housekeeping enzymes to master regulators of signal transduction. *FEBS J*, 280(2):346–78, 2013.

[4] N. K. Tonks. PTP1B: from the sidelines to the front lines! *FEBS Lett*, 546(1):140–8, 2003.

[5] S. Zhang and Z. Y. Zhang. PTP1B as a drug target: recent developments in PTP1B inhibitor discovery. *Drug Discov Today*, 12(9-10):373–81, 2007.

[6] S. Koren and I. G. Fantus. Inhibition of the protein tyrosine phosphatase PTP1B: potential therapy for obesity, insulin resistance and type-2 diabetes mellitus. *Best Pract Res Clin Endocrinol Metab*, 21(4):621–40, 2007.

[7] G. Moorhead (Ed.). *Protein Phosphatase Protocols*. Methods in Molecular Biology. Totowa, 2007.

[8] F. Sacco, L. Perfetto, L. Castagnoli, and G. Cesareni. The human phosphatase interactome: An intricate family portrait. *FEBS Lett*, 586(17):2732–9, 2012.

[9] N. K. Tonks. Protein tyrosine phosphatases: from genes, to function, to disease. *Nat Rev Mol Cell Biol*, 7(11):833–46, 2006.

[10] A. D. Pannifer, A. J. Flint, N. K. Tonks, and D. Barford. Visualization of the cysteinyl-phosphate intermediate of a protein-tyrosine phosphatase by x-ray crystallography. *J Biol Chem*, 273(17):10454–62, 1998.

[11] N. K. Tonks, C. D. Diltz, and E. H. Fischer. Purification of the major protein-tyrosine-phosphatases of human placenta. *J Biol Chem*, 263(14):6722–30, 1988.

[12] A. P. Combs. Recent advances in the discovery of competitive protein tyrosine phosphatase 1B inhibitors for the treatment of diabetes, obesity, and cancer. *J Med Chem*, 53(6):2333–44, 2010.

[13] J. C. Byon, A. B. Kusari, and J. Kusari. Protein-tyrosine phosphatase-1B acts as a negative regulator of insulin signal transduction. *Mol Cell Biochem*, 182(1-2):101–8, 1998.

[14] N. Dube and M. L. Tremblay. Involvement of the small protein tyrosine phosphatases TC-PTP and PTP1B in signal transduction and diseases: from diabetes, obesity to cell cycle, and cancer. *Biochim Biophys Acta*, 1754(1-2):108–17, 2005.

[15] T. O. Johnson, J. Ermolieff, and M. R. Jirousek. Protein tyrosine phosphatase 1B inhibitors for diabetes. *Nat Rev Drug Discov*, 1(9):696–709, 2002.

[16] M. Elchebly, P. Payette, E. Michaliszyn, W. Cromlish, S. Collins, A. L. Loy, D. Normandin, A. Cheng, J. Himms-Hagen, C. C. Chan, C. Ramachandran, M. J. Gresser, M. L. Tremblay, and B. P. Kennedy. Increased insulin sensitivity and obesity resistance in mice lacking the protein tyrosine phosphatase-1B gene. *Science*, 283(5407):1544–8, 1999.

[17] L. D. Klaman, O. Boss, O. D. Peroni, J. K. Kim, J. L. Martino, J. M. Zabolotny, N. Moghal, M. Lubkin, Y. B. Kim, A. H. Sharpe, A. Stricker-Krongrad, G. I. Shulman, B. G. Neel, and B. B. Kahn. Increased energy expenditure, decreased adiposity, and tissue-specific insulin sensitivity in protein-tyrosine phosphatase 1B-deficient mice. *Mol Cell Biol*, 20(15):5479–89, 2000.

[18] K. E. You-Ten, E. S. Muise, A. Itie, E. Michaliszyn, J. Wagner, S. Jothy, W. S. Lapp, and M. L. Tremblay. Impaired bone marrow microenvironment and immune function in t cell protein tyrosine phosphatase-deficient mice. *J Exp Med*, 186(5):683–93, 1997.

[19] D. A. Chistiakov and E. I. Chistiakova. T-cell protein tyrosine phosphatase: A role in inflammation and autoimmunity. *International Journal of Diabetes Mellitus*, (2):114–118, 2010.

[20] L. F. Iversen, K. B. Moller, A. K. Pedersen, G. H. Peters, A. S. Petersen, H. S. Andersen, S. Branner, S. B. Mortensen, and N. P. Moller. Structure determination of T cell protein-tyrosine phosphatase. *J Biol Chem*, 277(22):19982–90, 2002.

[21] L. Bialy and H. Waldmann. Inhibitors of protein tyrosine phosphatases: next-generation drugs? *Angew Chem Int Ed Engl*, 44(25):3814–39, 2005.

[22] Jr. Burke, T. R., B. Ye, X. Yan, S. Wang, Z. Jia, L. Chen, Z. Y. Zhang, and D. Barford. Small molecule interactions with protein-tyrosine phosphatase PTP1B and their use in inhibitor design. *Biochemistry*, 35(50):15989–96, 1996.

[23] Y. A. Puius, Y. Zhao, M. Sullivan, D. S. Lawrence, S. C. Almo, and Z. Y. Zhang. Identification of a second aryl phosphate-binding site in protein-tyrosine phosphatase 1B: a paradigm for inhibitor design. *Proc Natl Acad Sci U S A*, 94(25): 13420–5, 1997.

[24] J. P. Sun, A. A. Fedorov, S. Y. Lee, X. L. Guo, K. Shen, D. S. Lawrence, S. C. Almo, and Z. Y. Zhang. Crystal structure of PTP1B complexed with a potent and selective bidentate inhibitor. *J Biol Chem*, 278(14):12406–14, 2003.

[25] L. F. Iversen, H. S. Andersen, S. Branner, S. B. Mortensen, G. H. Peters, K. Norris, O. H. Olsen, C. B. Jeppesen, B. F. Lundt, W. Ripka, K. B. Moller, and N. P. Moller. Structure-based design of a low molecular weight, nonphosphorus, nonpeptide, and highly selective inhibitor of protein-tyrosine phosphatase 1B. *J Biol Chem*, 275(14):10300–7, 2000.

[26] G. Liu, Z. Xin, H. Liang, C. Abad-Zapatero, P. J. Hajduk, D. A. Janowick, B. G. Szczepankiewicz, Z. Pei, C. W. Hutchins, S. J. Ballaron, M. A. Stashko, T. H. Lubben, C. E. Berg, C. M. Rondinone, J. M. Trevillyan, and M. R. Jirousek. Selective protein tyrosine phosphatase 1B inhibitors: targeting the second phosphotyrosine binding site with non-carboxylic acid-containing ligands. *J Med Chem*, 46 (16):3437–40, 2003.

[27] B. G. Szczepankiewicz, G. Liu, P. J. Hajduk, C. Abad-Zapatero, Z. Pei, Z. Xin, T. H. Lubben, J. M. Trevillyan, M. A. Stashko, S. J. Ballaron, H. Liang, F. Huang, C. W.

Hutchins, S. W. Fesik, and M. R. Jirousek. Discovery of a potent, selective protein tyrosine phosphatase 1B inhibitor using a linked-fragment strategy. *J Am Chem Soc*, 125(14):4087–96, 2003.

[28] C. K. Lau, C. I. Bayly, J. Y. Gauthier, C. S. Li, M. Therien, E. Asante-Appiah, W. Cromlish, Y. Boie, F. Forghani, S. Desmarais, Q. Wang, K. Skorey, D. Waddleton, P. Payette, C. Ramachandran, B. P. Kennedy, and G. Scapin. Structure based design of a series of potent and selective non peptidic PTP-1B inhibitors. *Bioorg Med Chem Lett*, 14(4):1043–8, 2004.

[29] X. Chen, Q. Gan, C. Feng, X. Liu, and Q. Zhang. Virtual screening of novel and selective inhibitors of protein tyrosine phosphatase 1B over t-cell protein tyrosine phosphatase using a bidentate inhibition strategy. *J Chem Inf Model*, 2018.

[30] E. Black, J. Breed, A. L. Breeze, K. Embrey, R. Garcia, T. W. Gero, L. Godfrey, P. W. Kenny, A. D. Morley, C. A. Minshull, A. D. Pannifer, J. Read, A. Rees, D. J. Russell, D. Toader, and J. Tucker. Structure-based design of protein tyrosine phosphatase-1B inhibitors. *Bioorg Med Chem Lett*, 15(10):2503–7, 2005.

[31] A. P. Combs, E. W. Yue, M. Bower, P. J. Ala, B. Wayland, B. Douty, A. Takvorian, P. Polam, Z. Wasserman, W. Zhu, M. L. Crawley, J. Pruitt, R. Sparks, B. Glass, D. Modi, E. McLaughlin, L. Bostrom, M. Li, L. Galya, K. Blom, M. Hillman, L. Gonneville, B. G. Reid, M. Wei, M. Becker-Pasha, R. Klabe, R. Huber, Y. Li, G. Hollis, T. C. Burn, R. Wynn, P. Liu, and B. Metcalf. Structure-based design and discovery of protein tyrosine phosphatase inhibitors incorporating novel isothiazolidinone heterocyclic phosphotyrosine mimetics. *J Med Chem*, 48(21):6544–8, 2005.

[32] E. W. Yue, B. Wayland, B. Douty, M. L. Crawley, E. McLaughlin, A. Takvorian, Z. Wasserman, M. J. Bower, M. Wei, Y. Li, P. J. Ala, L. Gonneville, R. Wynn, T. C. Burn, P. C. Liu, and A. P. Combs. Isothiazolidinone heterocycles as inhibitors of protein tyrosine phosphatases: synthesis and structure-activity relationships of a peptide scaffold. *Bioorg Med Chem*, 14(17):5833–49, 2006.

[33] Y. Du, Y. Zhang, H. Ling, Q. Li, and J. Shen. Discovery of novel high potent and cellular active ADC type PTP1B inhibitors with selectivity over TC-PTP via modification interacting with C site. *Eur J Med Chem*, 144:692–700, 2018.

[34] D. Sydow. *Dynophores: Novel Dynamic Pharmacophores*. Humboldt-Universität zu Berlin, Lebenswissenschaftliche Fakultät, 2015.

[35] A. Bock, M. Bermudez, F. Krebs, C. Matera, B. Chirinda, D. Sydow, C. Dallanoce, U. Holzgrabe, M. De Amici, M. J. Lohse, G. Wolber, and K. Mohr. Ligand binding ensembles determine graded agonist efficacies at a G protein-coupled receptor. *J Biol Chem*, 291(31):16375–89, 2016.

[36] J. Mortier, J. R. C. Prevost, D. Sydow, S. Teuchert, C. Omieczynski, M. Bermudez, R. Frederick, and G. Wolber. Arginase structure and inhibition: Catalytic site plasticity reveals new modulation possibilities. *Sci Rep*, 7(1):13616, 2017.

[37] G. Sliwoski, S. Kothiwale, J. Meiler, and Jr. Lowe, E. W. Computational methods in drug discovery. *Pharmacol Rev*, 66(1):334–95, 2014.

[38] I. Muegge, A. Bergner, and J. M. Kriegl. Computer-aided drug design at Boehringer Ingelheim. *J Comput Aided Mol Des*, 31(3):275–285, 2017.

[39] F. K. Brown, E. C. Sherer, S. A. Johnson, M. K. Holloway, and B. S. Sherborne. The evolution of drug design at Merck Research Laboratories. *J Comput Aided Mol Des*, 31(3):255–266, 2017.

[40] C. Hansch and T. Fujita. p-s-p analysis. a method for the correlation of biological activity and chemical structure. *Journal of the American Chemical Society*, 86(8): 1616–1626, 1964.

[41] D. E. Clark. What has computer-aided molecular design ever done for drug discovery? *Expert Opin Drug Discov*, 1(2):103–10, 2006.

[42] *Maestro*. Schrödinger LLC, 2018.

[43] *Molecular Operating Environment (MOE)*. Chemical Computing Group ULC, 1010 Sherbooke St. West, Suite #910, Montreal, QC, Canada, H3A 2R7, 2018.

[44] S. Cross and G. Cruciani. Molecular fields in drug discovery: getting old or reaching maturity? *Drug Discov Today*, 15(1-2):23–32, 2010.

[45] *Standardizer*. ChemAxon, 2018.

[46] H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. N. Shindyalov, and P. E. Bourne. The Protein Data Bank. *Nucleic Acids Res*, 28(1): 235–42, 2000.

[47] Ardell D.H. Illergård, K. and A. Elofsson. Structure is three to ten times more conserved than sequence –a study of structural response in protein cores. *Proteins*, (77):499–508, 2009.

[48] A. Hillisch, L. F. Pineda, and R. Hilgenfeld. Utility of homology models in the drug discovery process. *Drug Discov Today*, 9(15):659–69, 2004.

[49] N. Eswar, B. Webb, M. A. Marti-Renom, M.S. Madhusudhan, D. Eramian, M. Shen, U. Pieper, and A. Sali. Comparative protein structure modeling using Modeller. *Current Protocols in Bioinformatics*, 28(5.6.1-5.6.32), 2014.

[50] Lawrence A. Kelley, Stefans Mezulis, Christopher M. Yates, Mark N. Wass, and Michael J. E. Sternberg. The Phyre2 web portal for protein modeling, prediction and analysis. *Nature Protocols*, 10:845, 2015.

[51] M. Biasini, S. Bienert, A. Waterhouse, K. Arnold, G. Studer, T. Schmidt, F. Kiefer, T. G. Cassarino, M. Bertoni, L. Bordoli, and T. Schwede. SWISS-MODEL: modelling protein tertiary and quaternary structure using evolutionary information. *Nucleic Acids Research*, 42(Web Server issue):W252–W258, 2014.

[52] T. Schwede. Protein modeling: what happened to the "protein structure gap"? *Structure*, 21(9):1531–40, 2013.

[53] P. A. Sousa, S. F.and Fernandes and M. J. Ramos. Protein-ligand docking: Current status and future challenges. *Proteins: Structure, Function, and Bioinformatics*, 65(1):15–26, 2006.

[54] G. Jones, P. Willett, R. C. Glen, A. R. Leach, and R. Taylor. Development and validation of a genetic algorithm for flexible docking. *J Mol Biol*, 267(3):727–48, 1997.

[55] D. B. Kitchen, H. Decornez, J. R. Furr, and J. Bajorath. Docking and scoring in virtual screening for drug discovery: methods and applications. *Nat Rev Drug Discov*, 3(11):935–49, 2004.

[56] R. T. Kroemer. Structure-based drug design: docking and scoring. *Curr Protein Pept Sci*, 8(4):312–28, 2007.

[57] G. L. Warren, C. W. Andrews, A. M. Capelli, B. Clarke, J. LaLonde, M. H. Lambert, M. Lindvall, N. Nevins, S. F. Semus, S. Senger, G. Tedesco, I. D. Wall, J. M. Woolven, C. E. Peishoff, and M. S. Head. A critical assessment of docking programs and scoring functions. *J Med Chem*, 49(20):5912–31, 2006.

[58] B. Q. Wei, L. H. Weaver, A. M. Ferrari, B. W. Matthews, and B. K. Shoichet. Testing a flexible-receptor docking algorithm in a model binding site. *J Mol Biol*, 337(5): 1161–82, 2004.

[59] G. M. Morris, W. Huey, R.and Lindstrom, M. F. Sanner, R. K. Belew, D. S. Goodsell, and A. J. Olson. AutoDock4 and AutoDockTools4: Automated docking with selective receptor flexibility. *Journal of computational chemistry*, 30(16):2785–2791, 2009.

[60] S. Forli, M. E. Huey, R.and Pique, M. Sanner, D. S. Goodsell, and A. J. Olson. Computational protein-ligand docking and virtual drug screening with the AutoDock suite. *Nature protocols*, 11(5):905–919, 2016.

[61] R. G. Coleman, M. Carchia, T. Sterling, J. J. Irwin, and B. K. Shoichet. Ligand pose and orientational sampling in molecular docking. *PLOS ONE*, 8(10):e75992, 2013.

[62] M. Rarey, T. Kramer, B.and Lengauer, and G. Klebe. A fast flexible docking method using an incremental construction algorithm. *Journal of Molecular Biology*, 261 (3):470–489, 1996.

[63] R. A. Friesner, R. B. Murphy, M. P. Repasky, L. L. Frye, J. R. Greenwood, T. A. Halgren, P. C. Sanschagrin, and D. T. Mainz. Extra precision Glide: Docking and scoring incorporating a model of hydrophobic enclosure for protein-ligand complexes. *Journal of Medicinal Chemistry*, 49(21):6177–6196, 2006.

[64] M. D. Cummings, R. L. DesJarlais, A. C. Gibbs, V. Mohan, and E. P. Jaeger. Comparison of automated docking programs as virtual screening tools. *J Med Chem*, 48(4):962–76, 2005.

[65] C. Bissantz, B. Kuhn, and M. Stahl. A medicinal chemist's guide to molecular interactions. *J Med Chem*, 53(14):5061–84, 2010.

[66] A. Ganesan, M. L. Coote, and K. Barakat. Molecular dynamics-driven drug discovery: leaping forward with confidence. *Drug Discov Today*, 22(2):249–269, 2017.

[67] A. Hospital, J. R. Goni, M. Orozco, and J. L. Gelpi. Molecular dynamics simulations: advances and applications. *Adv Appl Bioinform Chem*, 8:37–47, 2015.

[68] M. P. Allen. Introduction to molecular dynamics simulation. volume 23 of *NIC Series*. John von Neumann Institute for Computing, 2004.

[69] R. Salomon-Ferrer, D.A. Case, and R. C. Walker. An overview of the Amber biomolecular simulation package. *Wiley Interdisciplinary Reviews: Computational Molecular Science*, 3(2):198–210, 2013.

[70] S. Pronk, S. Páll, R. Schulz, P. Larsson, P. Bjelkmar, R. Apostolov, M. R. Shirts, J. C. Smith, P. M. Kasson, D. van der Spoel, B. Hess, and E. Lindahl. GROMACS 4.5: a high-throughput and highly parallel open source molecular simulation toolkit. *Bioinformatics*, 29(7):845–854, 2013.

[71] K. J. Bowers, D. E. Chow, H. Xu, R. O. Dror, M. P. Eastwood, B. A. Gregersen, J. L. Klepeis, I. Kolossvary, M. A. Moraes, F. D. Sacerdoti, J. K. Salmon, Y. Shan, and D. E. Shaw. Scalable algorithms for molecular dynamics simulations on commodity clusters. In *SC 2006 Conference, Proceedings of the ACM/IEEE*, pages 43–43, 2006.

[72] J.C. Phillips, W. Braun, R.and Wang, J. Gumbart, E. Tajkhorshid, E. Villa, C. Chipot, R. D. Skeel, L. Kalé, and K. Schulten. Scalable molecular dynamics with NAMD. *Journal of computational chemistry*, 26(16):1781–1802, 2005.

[73] P. Eastman, M. S. Friedrichs, J. D. Chodera, R. J. Radmer, C. M. Bruns, J.P. Ku, K. A. Beauchamp, T. J. Lane, D. Wang, L.-P.and Shukla, T. Tye, M. Houston, T. Stich, C. Klein, M. R. Shirts, and V. S. Pande. OpenMM 4: A reusable, extensible, hardware independent library for high performance molecular simulation. *Journal of Chemical Theory and Computation*, 9(1):461–469, 2013.

[74] F. R. Salsbury. Molecular dynamics simulations of protein dynamics and their relevance to drug discovery. *Current opinion in pharmacology*, 10(6):738–744, 2010.

[75] H.-J. Böhm and G. Schneider. *Protein-Ligand Interactions: From Molecular Recognition to Drug Design.* WILEY-VCH, Weinheim, 2003.

[76] D. L. Mobley and K. A. Dill. Binding of small-molecule ligands to proteins: "what you see" is not always "what you get". *Structure*, 17(4):489–98, 2009.

[77] A. Artese, S. Alcaro, F. Moraca, R. Reina, M. Ventura, G. Costantino, A. R. Beccari, and F. Ortuso. State-of-the-art and dissemination of computational tools for drug-design purposes: a survey among italian academics and industrial institutions. *Future Med Chem*, 5(8):907–27, 2013.

[78] E. Carosati and G. Sciabola, S.and Cruciani. Hydrogen bonding interactions of covalently bonded fluorine atoms: From crystallographic data to a new angular function in the grid force field. *Journal of Medicinal Chemistry*, 47(21):5114–5125, 2004.

[79] P. Goodford. The basic principles of GRID. In *Molecular Interaction Fields*, pages 1–25. Wiley-VCH Verlag GmbH & Co. KGaA, 2005.

[80] G. Wolber and T. Langer. LigandScout: 3-D pharmacophores derived from protein-bound ligands and their use as virtual screening filters. *Journal of Chemical Information and Modeling*, 45(1):160–169, 2005.

[81] G. Wolber, A. A. Dornhofer, and T. Langer. Efficient overlay of small organic molecules using 3D pharmacophores. *J Comput Aided Mol Des*, 20(12):773–788, 2006.

[82] C. G. Wermuth, C. R. Ganellin, P. Lindberg, and L. A. Mitscher. Glossary of terms used in medicinal chemistry. *Pure & Appl. Chem.*, 70(5):15, 1998.

[83] U. Perricone, M. Wieder, T. Seidel, T. Langer, A. Padova, A. M. Almerico, and M. Tutone. A molecular dynamics-shared pharmacophore approach to boost early-enrichment virtual screening: A case study on peroxisome proliferator-activated receptor alpha. *ChemMedChem*, 12(16):1399–1407, 2017.

[84] T. Seidel, G. Ibis, F. Bendix, and G. Wolber. Strategies for 3D pharmacophore-based virtual screening. *Drug Discovery Today: Technologies*, 7(4):e221–e228, 2010.

[85] *CATALYST, Discovery Studio Modeling Environment.* Dassault Systèmes BIOVIA, 2016.

[86] S. L. Dixon, A. M. Smondyrev, E. H. Knoll, S. N. Rao, D. E. Shaw, and R. A. Friesner. PHASE: a new engine for pharmacophore perception, 3D QSAR model development, and 3D database screening: 1. methodology and preliminary results. *J Comput Aided Mol Des*, 20(10-11):647–671, 2006.

[87] S. Cross, M. Baroni, L. Goracci, and G. Cruciani. GRID-based three-dimensional pharmacophores I: FLAPpharm, a novel approach for pharmacophore elucidation. *Journal of Chemical Information and Modeling*, 52(10):2587–2598, 2012.

[88] M. P. A. Sanders, R. McGuire, L. Roumen, I. J. P. de Esch, J. P. G. de Vlieg, J.and Klomp, and C. de Graaf. From the protein's perspective: the benefits and challenges of protein structure-based pharmacophore modeling. *MedChem-Comm*, 3(1):28–38, 2012.

[89] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, 2018.

[90] J. P. Changeux and S. Edelstein. Conformational selection or induced fit? 50 years of debate resolved. *F1000 Biol Rep*, 3:19, 2011.

[91] A. D. Vogt and E. Di Cera. Conformational selection is a dominant mechanism of ligand binding. *Biochemistry*, 52(34):5723–9, 2013.

[92] C. E. Chang, W. Chen, and M. K. Gilson. Ligand configurational entropy and protein binding. *Proc Natl Acad Sci U S A*, 104(5):1534–9, 2007.

[93] G. Klebe. *Wirkstoffdesign - Entwurf und Wirkung von Arzneistoffen*. Springer Spektrum, 2009.

[94] J. D. Durrant, C. A. de Oliveira, and J. A. McCammon. POVME: an algorithm for measuring binding-pocket volumes. *J Mol Graph Model*, 29(5):773–6, 2011.

[95] J. D. Durrant, L. Votapka, J. Sorensen, and R. E. Amaro. POVME 2.0: An enhanced tool for determining pocket shape and volume characteristics. *J Chem Theory Comput*, 10(11):5047–5056, 2014.

[96] F. Paul, C. Wehmeyer, E. T. Abualrous, H. Wu, M. D. Crabtree, J. Schöneberg, J. Clarke, C. Freund, T. R. Weikl, and F. Noé. Protein-peptide association kinetics beyond the seconds timescale from atomistic simulations. *Nat Commun*, 8 (1):1095, 2017.

[97] A. P. Montgomery, D. Skropeta, and H. Yu. Transition state-based ST6Gal I inhibitors: Mimicking the phosphodiester linkage with a triazole or carbamate through an enthalpy-entropy compensation. *Sci Rep*, 7(1):14428, 2017.

[98] J. Shao, S. W. Tanner, N. Thompson, and T. E. Cheatham. Clustering molecular dynamics trajectories: 1. characterizing the performance of different clustering algorithms. *J Chem Theory Comput*, 3(6):2312–34, 2007.

[99] L. Kaufman and P. J. Rousseeuw. Introduction. In *Finding Groups in Data*, pages 1–67. John Wiley & Sons, Inc., 1990.

[100] F. Murtagh and P. Legendre. Ward's hierarchical agglomerative clustering method: Which algorithms implement Ward's criterion? *Journal of Classification*, 31(3):274–295, 2014.

[101] F. Holmes. Comparison of distance measures in cluster analysis with dichotomous data. *Journal of Data Science*, 3(1):85–100, 2005.

[102] V. Consonni and R. Todeschini. New similarity coefficients for binary data. *MATCH Communications in Mathematical and in Computer Chemistry*, 68(2): 581–592, 2012.

[103] R. Tibshirani, G. Walther, and T. Hastie. Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(2):411–423, 2001.

[104] F. Sievers, D. Wilm, A.and Dineen, T. J. Gibson, K. Karplus, W. Li, R. Lopez, H. McWilliam, J. Remmert, M.and Söding, J. D. Thompson, and D. G. Higgins. Fast, scalable generation of high-quality protein multiple sequence alignments using Clustal Omega. *Molecular Systems Biology*, 7(1), 2011.

[105] X. Li, L. Wang, and D. Shi. The design strategy of selective PTP1B inhibitors over TCPTP. *Bioorg Med Chem*, 24(16):3343–52, 2016.

[106] S. R. Klopfenstein, A. G. Evdokimov, N. T. Colson, A.-O.and Fairweather, J. J. Neuman, M.B. Maier, J. L. Gray, G. S. Gerwe, G. E. Stake, B. W. Howard, J. A. Farmer, M. E. Pokross, T. R. Downs, B. Kasibhatla, and K. G. Peters. 1,2,3,4-tetrahydroisoquinolinyl sulfamic acids as phosphatase PTP1B inhibitors. *Bioorganic & Medicinal Chemistry Letters*, 16(6):1574–1578, 2006.

[107] A. P. Bento, A. Gaulton, A. Hersey, L. J. Bellis, J. Chambers, M. Davies, F. A. Kruger, Y. Light, L. Mak, S. McGlinchey, M. Nowotka, G. Papadatos, R. Santos, and J. P. Overington. The ChEMBL bioactivity database: an update. *Nucleic Acids Res*, 42 (Database issue):D1083–90, 2014.

[108] Z. Xin, T. K. Oost, C. Abad-Zapatero, P. J. Hajduk, Z. Pei, B. G. Szczepankiewicz, C. W. Hutchins, S. J. Ballaron, M. A. Stashko, T. Lubben, J. M. Trevillyan, M. R.

Jirousek, and G. Liu. Potent, selective inhibitors of protein tyrosine phosphatase 1B. *Bioorg Med Chem Lett*, 13(11):1887–90, 2003.

[109] A. F. Moretto, S. J. Kirincich, W. X. Xu, M. J. Smith, Z. K. Wan, D. P. Wilson, B. C. Follows, E. Binnun, D. Joseph-McCarthy, K. Foreman, D. V. Erbe, Y. L. Zhang, S. K. Tam, S. Y. Tam, and J. Lee. Bicyclic and tricyclic thiophenes as protein tyrosine phosphatase 1B inhibitors. *Bioorg Med Chem*, 14(7):2162–77, 2006.

[110] The UniProt Consortium. UniProt: the universal protein knowledgebase. *Nucleic Acids Res*, 45(D1):D158–D169, 2017.

[111] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *J. ACM*, 45(6):891–923, 1998.

[112] L. Pace (Ed.). The "new" statistics: Resampling and bootstrapping. In *Beginning R: An Introduction to Statistical Programming*, pages 257–268. Apress, Berkeley, CA, 2012.

[113] C. Barillari, G. Marcou, and D. Rognan. Hot-spots-guided receptor-based pharmacophores (HS-Pharm): A knowledge-based approach to identify ligand-anchoring atoms in protein cavities and prioritize structure-based pharmacophores. *Journal of Chemical Information and Modeling*, 48(7):1396–1410, 2008.

[114] N. A. Meanwell. Fluorine and fluorinated motifs in the design and application of bioisosteres for drug design. *J Med Chem*, 2018.

[115] F. Sievers, A. Wilm, D. Dineen, T. J. Gibson, K. Karplus, W. Li, R. Lopez, H. McWilliam, M. Remmert, J. Soding, J. D. Thompson, and D. G. Higgins. Fast, scalable generation of high-quality protein multiple sequence alignments using clustal omega. *Mol Syst Biol*, 7:539, 2011.

[116] Y. A. Puius, Y. Zhao, M. Sullivan, D. S. Lawrence, S. C. Almo, and Z. Y. Zhang. Identification of a second aryl phosphate-binding site in protein-tyrosine phosphatase 1B: a paradigm for inhibitor design. *Proc Natl Acad Sci USA*, 94(25): 13420–5, 1997.

[117] J. Montalibet, K. Skorey, D. McKay, G. Scapin, E. Asante-Appiah, and B. P. Kennedy. Residues distant from the active site influence protein-tyrosine phosphatase 1B inhibitor binding. *J Biol Chem*, 281(8):5258–66, 2006.

[118] M. S. Choy, Y. Li, Lesf Machado, M. B. A. Kunze, C. R. Connors, X. Wei, K. Lindorff-Larsen, R. Page, and W. Peti. Conformational rigidity and protein dynamics at distinct timescales regulate PTP1B activity and allostery. *Mol Cell*, 65(4):644–658 e5, 2017.

[119] Z. Xin, G. Liu, C. Abad-Zapatero, Z. Pei, B. G. Szczepankiewicz, X. Li, T. Zhang, C. W. Hutchins, P. J. Hajduk, S. J. Ballaron, M. A. Stashko, T. H. Lubben, J. M. Trevillyan, and M. R. Jirousek. Identification of a monoacid-based, cell permeable, selective inhibitor of protein tyrosine phosphatase 1B. *Bioorg Med Chem Lett*, 13 (22):3947–50, 2003.

[120] G. Liu, Z. Xin, Z. Pei, P. J. Hajduk, C. Abad-Zapatero, C. W. Hutchins, H. Zhao, T. H. Lubben, S. J. Ballaron, D. L. Haasch, W. Kaszubska, C. M. Rondinone, J. M. Trevillyan, and M. R. Jirousek. Fragment screening and assembly: a highly efficient approach to a selective and cell active protein tyrosine phosphatase 1B inhibitor. *J Med Chem*, 46(20):4232–5, 2003.

[121] H. Zhao, G. Liu, Z. Xin, M. D. Serby, Z. Pei, B. G. Szczepankiewicz, P. J. Hajduk, C. Abad-Zapatero, C. W. Hutchins, T. H. Lubben, S. J. Ballaron, D. L. Haasch, W. Kaszubska, C. M. Rondinone, J. M. Trevillyan, and M. R. Jirousek. Isoxazole carboxylic acids as protein tyrosine phosphatase 1B (PTP1B) inhibitors. *Bioorganic & Medicinal Chemistry Letters*, 14(22):5543–5546, 2004.

[122] W. Humphrey, A. Dalke, and K. Schulten. VMD: visual molecular dynamics. *J Mol Graph*, 14(1):33–38, 1996.

[123] B. J. Grant, A. P. Rodrigues, K. M. ElSawy, J. A. McCammon, and L. S. Caves. Bio3d: an R package for the comparative analysis of protein structures. *Bioinformatics*, 22(21):2695–6, 2006.

A. Appendix

A.1. Modified Version of the POVME2 Tool

Highlighted lines were added to the original script.

```
# POVME 2.0.1 is released under the GNU General Public License (
    see http://www.gnu.org/licenses/gpl.html).
# If you have any questions, comments, or suggestions, please
    don't hesitate to contact me,
# Jacob Durrant, at jdurrant [at] ucsd [dot] edu.
#
# If you use POVME in your work, please cite Durrant, J. D., C.
    A. de Oliveira, et al.
#    (2011). "POVME: An algorithm for measuring binding-pocket
    volumes." J Mol Graph
#    Model 29(5): 773-776.

import math
import sys
import time
import numpy
import pymolecule
import gzip
import os
import shutil
import random
import multiprocessing
import platform

```

```
try: from cStringIO import StringIO
except: from StringIO import StringIO

from scipy.spatial.distance import cdist
from scipy.spatial.distance import pdist
from scipy.spatial.distance import squareform

version = "2.0.1"

def log(astr, parameters):
'''Output POVME statements, either to the screen or to a file

Arguments:
astr -- The string to output.
parameters -- The user-defined parameters.

'''

# Print the output to the screen.
print astr

# Save it to the output file as well.
try:
if parameters['CompressOutput'] == True: f = gzip.open(
    parameters['OutputFilenamePrefix'] + 'output.txt.gz', 'ab')
else: f = open(parameters['OutputFilenamePrefix'] + 'output.txt'
    , 'a')

f.write(astr + "\n")
f.close()
except: pass

class Multithreading():
"""A class for running calculations on multiple processors"""

```

```
results = []

def __init__(self, inputs, num_processors, task_class):
"""Launches a calculation on multiple processors

Arguments:
inputs -- A list, containing all the input required for the
    calculation
num_processors -- An integer, the requested number of processors
    to use
task_class -- An class, the class governing what calculations
    will be run on a given thread

Returns:
Nothing, though the objects self.results list is populated with
    the calculation results

"""

self.results = []

if num_processors != 1 and (platform.system().upper()[:3] == "
    WIN" or "NT" in platform.system().upper()): # If it's windows
    , you can only use one processor.
print "WARNING:␣Use␣of␣multiple␣processors␣is␣not␣supported␣in␣
    Windows.␣Proceeding␣with␣one␣processor..."
num_processors = 1

if num_processors == 1: # so just running on 1 processor,
    perhaps under windows
single_thread = task_class()
single_thread.total_num_tasks = len(inputs)

single_thread.results = []
for item in inputs: single_thread.value_func(item, None)
```

```

self.results = single_thread.results

else: # so it actually is running on multiple processors

cpu_count = 1
cpu_count = multiprocessing.cpu_count()

# first, if num_processors <= 0, determine the number of
    processors to use programatically
if num_processors <= 0: num_processors = cpu_count

# reduce the number of processors if too many have been
    specified
if len(inputs) < num_processors: num_processors = len(inputs)

if len(inputs) == 0: # if there are no inputs, there's nothing
    to do.
self.results = []
return

# now, divide the inputs into the appropriate number of
    processors
inputs_divided = {}
for t in range(num_processors): inputs_divided[t] = []

for t in range(0, len(inputs), num_processors):
for t2 in range(num_processors):
index = t + t2
if index < len(inputs): inputs_divided[t2].append(inputs[index])

# now, run each division on its own processor
running = multiprocessing.Value('i', num_processors)
mutex = multiprocessing.Lock()

```

```
arrays = []
threads = []
for i in range(num_processors):
athread = task_class()
athread.total_num_tasks = len(inputs)

threads.append(athread)
arrays.append(multiprocessing.Array('i',[0, 1]))

results_queue = multiprocessing.Queue() # to keep track of the
    results

processes = []
for i in range(num_processors):
p = multiprocessing.Process(target=threads[i].runit, args=(
    running, mutex, results_queue, inputs_divided[i]))
p.start()
processes.append(p)

while running.value > 0: is_running = 0 # wait for everything to
    finish

# compile all results into one list
for thread in threads:
chunk = results_queue.get()
self.results.extend(chunk)

class MultithreadingTaskGeneral:
"""A parent class of others that governs what calculations are
    run on each thread"""

results = []

def runit(self, running, mutex, results_queue, items):
"""Launches the calculations on this thread
```

```

Arguments:
running −− A multiprocessing.Value object
mutex −− A multiprocessing.Lock object
results_queue −− A multiprocessing.Queue() object for storing the calculation output
items −− A list, the input data required for the calculation

"""

for item in items: self.value_func(item, results_queue)

mutex.acquire()
running.value −= 1
mutex.release()
results_queue.put(self.results)

def value_func(self, item, results_queue): # so overwriting this function
"""The definition that actually does the work.

Arguments:
item −− A list or tuple, the input data required for the calculation
results_queue −− A multiprocessing.Queue() object for storing the calculation output

"""

# input1 = item[0]
# input2 = item[1]
# input3 = item[2]
# input4 = item[3]
# input5 = item[4]
# input6 = item[5]

# use inputs to come up with a result, some_result

#self.results.append(some_result)

pass

class ConvexHull():
"""A class to handle convex-hull calculations"""

def get_seg_dict_num(self, seg_dict, seg_index):
"""seg_dict is a dictionary object that contains information
    about segments within the convex hull. The keys are 2x3
    tuples, which represent two ends of a segment in space. The
    values of seg_dict are the number of times a segment has been
     part of a triangle, either 1 or 2. (Zero times would mean
    that the segment doesn't exist in the dictionary yet). This
    function looks up and returns the value of a seg_index from
    seg_dict

Arguments:
seg_dict -- the dictionary of segment 2x3 tuples as keys,
    integers as values
seg_index -- the key of the dictionary member we are going to
    retrieve

Returns:
if seg_index exists in the keys of seg_dict, return the value.
    Otherwise, return 0

"""

if seg_index[0][0] > seg_index[1][0]: # we want the index with
    the greater x-value, so we don't get identical segments in
    the dictionary more than once
```

```
index = seg_index
else:
index = seg_index[::-1]

if index in seg_dict:
return seg_dict[index]
else:
return 0

def increment_seg_dict(self, seg_dict, seg_index):
"""seg_dict is a dictionary object that contains information
    about segments within the convex hull. The keys are 2x3
    tuples, which represent two ends of a segment in space. The
    values of seg_dict are the number of times a segment has been
     part of a triangle, either 1 or 2. (Zero times would mean
    that the segment doesn't exist in the dictionary yet). This
    function increments the values within seg_dict, or initiates
    them if they dont exist yet.

Arguments:
seg_dict -- the dictionary of segment 2x3 tuples as keys,
    integers as values
seg_index -- the key of the dictionary member we are going to
    increment

Returns:
None: the values of seg_dict are received and modified by
    reference
"""

if seg_index[0][0] > seg_index[1][0]: # we want the index with
    the greater x-value, so we don't get identical segments in
    the dictionary more than once
index = seg_index
else:
```

```
index = seg_index[::−1]

#"putting index:", index, "into seg_dict because", index[0][0], ">", index[1][0]

if index in seg_dict: # if the entry already exists in seg_dict
seg_dict[index] += 1 # increment
else:
seg_dict[index] = 1 # initiate with a value of 1 because it now exists on a triangle
return

def gift_wrapping_3d(self, raw_points):
"""Gift wrapping for 3d convex hull

Arguments:
raw_points −− A nx3 array of points, where each row corresponds to an x,y,z point coordinate

Returns:
A convex hull represented by a list of triangles. Each triangle is a 3x3 array, where each row is an x,y,z coordinate in space. The 3 rows describe the location of the 3 corners of the triangle. Each of the 3 points are arranged so that a cross product will point outwards from the hull


"""

n = numpy.shape(raw_points)[0] # number of points
point1 = raw_points[0] # take the first point
xaxis = numpy.array([1,0,0]) # create a ref vector pointing along x axis
maxx = raw_points[0][0] # initiate highest x value
points = [] # a list of tuples for easy dictionary lookup
```

```
seg_dict = {} # a dictionary that contains the number of
    triangles a seg is in

begintime = time.time()
for i in range(n): # find the n with the largest x value
point = tuple(raw_points[i])
points.append(point)
if point[0] > maxx:
maxx = point[0]
point1 = raw_points[i]
#print "find max x:", time.time() - begintime

best_dot = -1.0 # initiate dot relative to x-axis
point2 = numpy.array(raw_points[1]) # initiate best segment

# find first/best segment
begintime = time.time()
for i in range(n):
pointi = raw_points[i]
if numpy.array_equal(pointi, point1): continue
diff_vec = pointi - point1
diff_len = numpy.linalg.norm(diff_vec)

test_dot = numpy.dot(diff_vec/diff_len,xaxis)
if test_dot > best_dot:
best_dot = test_dot
point2 = pointi

#print "find first segment:", time.time() - begintime
point1 = tuple(point1)
point2 = tuple(point2)
ref_vec = xaxis

# now find the best triangle
triangles = []
```

```

seg_list = set([(point1, point2),])
norm_dict = {(point1,point2):xaxis}
self.increment_seg_dict( seg_dict, (point1,point2) )

counter = 0
first_time = True

begintime = time.time()
section1 = 0.0
section2 = 0.0
section3 = 0.0
while seg_list: # as long as there are unexplored edges of
    triangles in the hull...

counter += 1
seg = seg_list.pop() # take a segment out of the seg_list
tuple1 = seg[0] # the two ends of the segment
tuple2 = seg[1]
point1 = numpy.array(seg[0])
point2 = numpy.array(seg[1])
result = self.get_seg_dict_num( seg_dict, (seg[0],seg[1]) )

if result >= 2: # then we already have 2 triangles on this
    segment
continue # forget about drawing a triangle for this seg

ref_vec = norm_dict[(seg[0],seg[1])] # get the norm for a
    triangle that the segment is part of

best_dot_cross = -1.0
best_point = None

for i in range(n): # look at each point

```

```python
pointi = raw_points[i]
#if numpy.array_equal(pointi, point1) or numpy.array_equal(pointi, point2): continue # if we are trying one of the points that are point1 or point2
diff_vec1 = point2 - point1
#diff_len1 = numpy.linalg.norm(diff_vec1)
diff_vec2 = pointi - point2
#diff_len2 = numpy.linalg.norm(diff_vec2)

#test_cross = numpy.cross(diff_vec1/diff_len1,diff_vec2/diff_len2)
#test_cross = numpy.cross(diff_vec1,diff_vec2)
test_cross = numpy.array([diff_vec1[1]*diff_vec2[2]-diff_vec1[2]*diff_vec2[1], diff_vec1[2]*diff_vec2[0]-diff_vec1[0]*diff_vec2[2], diff_vec1[0]*diff_vec2[1]-diff_vec1[1]*diff_vec2[0]]) # cross product

test_cross_len = numpy.sqrt(test_cross[0]*test_cross[0] + test_cross[1]*test_cross[1] + test_cross[2]*test_cross[2]) # numpy.linalg.norm(test_cross) # get the norm of the cross product

if test_cross_len <= 0.0: continue
#test_cross_len_inv = 1 / test_cross_len
test_cross = test_cross / test_cross_len
dot_cross = numpy.dot(test_cross, ref_vec)
#dot_cross = test_cross[0]*ref_vec[0] + test_cross[1]*ref_vec[1] + test_cross[2]*ref_vec[2]
if dot_cross > best_dot_cross:
best_cross = test_cross
best_dot_cross = dot_cross
best_point = pointi
tuple3 = points[i]


```

```
point3 = best_point

if self.get_seg_dict_num( seg_dict, (tuple2,tuple1) ) > 2:
    continue
if self.get_seg_dict_num( seg_dict, (tuple3,tuple2) ) > 2:
    continue
if self.get_seg_dict_num( seg_dict, (tuple1,tuple3) ) > 2:
    continue

# now we have a triangle from point1 -> point2 -> point3
# must test each edge
if first_time:
self.increment_seg_dict( seg_dict, (tuple2,tuple1) )
seg_list.add((tuple2, tuple1))
norm_dict[(tuple2,tuple1)] = best_cross

self.increment_seg_dict( seg_dict, (tuple3,tuple2) )
seg_list.add((tuple3, tuple2))
norm_dict[(tuple3,tuple2)] = best_cross

self.increment_seg_dict( seg_dict, (tuple1,tuple3) )
seg_list.add((tuple1, tuple3))
norm_dict[(tuple1,tuple3)] = best_cross

triangles.append((numpy.array(tuple1),numpy.array(tuple2),numpy.
    array(tuple3)))

first_time = False

#print "find all triangles:", time.time() - begintime

#print "section1:", section1
#print "section2:", section2
#print "section3:", section3
return triangles
```

```

def akl_toussaint(self, points):
"""The Akl–Toussaint Heuristic. Given a set of points, this definition will create an octahedron whose corners are the extremes in x, y, and z directions. Every point within this octahedron will be removed because they are not part of the convex hull. This causes any expected running time for a convex hull algorithm to be reduced to linear time.

Arguments:
points –– An nx3 array of x,y,z coordinates

Returns:
All members of original set of points that fall outside the Akl–Toussaint octahedron

"""

x_high = (−1e99,0,0); x_low = (1e99,0,0); y_high = (0,−1e99,0); y_low = (0,1e99,0); z_high = (0,0,−1e99); z_low = (0,0,1e99)


for point in points: # find the corners of the octahedron
if point[0] > x_high[0]: x_high = point
if point[0] < x_low[0]: x_low = point
if point[1] > y_high[1]: y_high = point
if point[1] < y_low[1]: y_low = point
if point[2] > z_high[2]: z_high = point
if point[2] < z_low[2]: z_low = point

octahedron = [ # define the triangles of the surfaces of the octahedron
numpy.array((x_high,y_high,z_high)),
numpy.array((x_high,z_low,y_high)),
numpy.array((x_high,y_low,z_low)),
numpy.array((x_high,z_high,y_low)),
numpy.array((x_low,y_low,z_high)),
numpy.array((x_low,z_low,y_low)),
numpy.array((x_low,y_high,z_low)),
numpy.array((x_low,z_high,y_high)),
]
new_points = [] # everything outside of the octahedron
for point in points: # now check to see if a point is inside or
    outside the octahedron
outside = self.outside_hull(point, octahedron, epsilon=-1.0e-5)
if outside:
new_points.append(point)

return numpy.array(new_points) # convert back to an array

def outside_hull(self, our_point, triangles, epsilon=1.0e-5):
"""Given the hull as defined by a list of triangles, this
    definition will return whether a point is within these or not
    .

Arguments:
our_point -- an x,y,z array that is being tested to see whether
    it exists inside the hull or not
triangles -- a list of triangles that define the hull
epsilon -- needed for imprecisions in the floating-point
    operations.

Returns:
True if our_point exists outside of the hull, False otherwise

"""

our_point = numpy.array(our_point) # convert it to an numpy.
    array
for triangle in triangles:
```

```
rel_point = our_point - triangle[0] # vector from triangle
    corner 0 to point
vec1 = triangle[1] - triangle[0] # vector from triangle corner 0
     to corner 1
vec2 = triangle[2] - triangle[1] # vector from triangle corner 1
     to corner 2
our_cross = numpy.cross(vec1, vec2) # cross product between vec1
     and vec2
our_dot = numpy.dot(rel_point,our_cross) # dot product to
    determine whether cross is point inward or outward
if numpy.dot(rel_point,our_cross) > epsilon: # if the dot is
    greater than 0, then its outside
return True

return False

def unique_rows(a):
"""Identifies unique points (rows) in an array of points.

Arguments:
a -- A nx3 numpy.array representing 3D points.

Returns:
A nx2 numpy.array containing the 3D points that are unique.

"""

a[a == -0.0] = 0.0
b = numpy.ascontiguousarray(a).view(numpy.dtype((numpy.void, a.
    dtype.itemsize * a.shape[1])))
unique_a = numpy.unique(b).view(a.dtype).reshape(-1, a.shape[1])

return unique_a

def create_pdb_line(numpy_array, index, resname, letter):
```

```
"""Create a string formatted according to the PDB standard.

Arguments:
numpy_array -- A 1x3 numpy.array representing a 3D point.
letter -- A string, the atom name/chain/etc to use for the
    output.

Returns:
A string, formatted according to the PDB standard.

"""

if len(numpy_array) == 2: numpy_array = numpy.array([numpy_array
    [0], numpy_array[1], 0.0])
if numpy_array.shape == (1, 3): numpy_array = numpy_array[0]

output = "ATOM "
output = output + str(index % 999999).rjust(6) + letter.rjust(5)
     + resname.rjust(4) + letter.rjust(2) + str(index % 9999).
    rjust(4)
output = output + ("%.3f" % numpy_array[0]).rjust(12)
output = output + ("%.3f" % numpy_array[1]).rjust(8)
output = output + ("%.3f" % numpy_array[2]).rjust(8)
output = output + letter.rjust(24)

return output

def numpy_to_pdb(narray, letter, resname=""):
"""Create a string formatted according to the PDB standard.

Arguments:
narray -- A nx3 numpy.array representing a 3D point.
letter -- A string, the atom name/chain/etc to use for the
    output.

```

```
Returns:
(Optionally) A string, formatted according to the PDB standard.

"""

if len(narray.flatten()) == 3:
return create_pdb_line(narray, 1, "AAA", letter) + "\n"
else:
if resname == "":
letters = ["A", "B", "C", "D", "E", "F", "G", "H", "I", "J", "K"
    , "L", "M", "N", "O", "P", "Q", "R", "S", "T", "U", "V", "W",
     "X", "Y", "Z"]
resnames = []
for l1 in letters:
for l2 in letters:
for l3 in letters:
resnames.append(l1+l2+l3)
resnames.remove("XXX") # because this is reserved for empty
    atoms
else:
resnames = [resname]

t = ""
for i, item in enumerate(narray): t = t + create_pdb_line(item,
    i+1, resnames[i % len(resnames)], letter) + "\n"
return t

def dx_freq(freq_mat, parameters):
'''
Generates a DX file that records the frequency that a volume
    element is open

Arguments:
freq_mat -- a Nx4 matrix, where the first 3 columns are the x,y,
    z coords of the point, and the 4th column is the frequency of
```

```
        emptiness for that point in space

'''

header_template = """# Data from POVME 2.0.1
#
# FREQUENCY (unitless)
#
object 1 class gridpositions counts %d %d %d
origin %8.6e %8.6e %8.6e
delta %8.6e 0.000000e+00 0.000000e+00
delta 0.000000e+00 %8.6e 0.000000e+00
delta 0.000000e+00 0.000000e+00 %8.6e
object 2 class gridconnections counts %d %d %d
object 3 class array type double rank 0 items %d data follows
"""

footer_template = """
attribute "dep" string "positions"
object "regular positions regular connections" class field
component "positions" value 1
component "connections" value 2
component "data" value 3"""

# 1. Sort the points into the proper order for a dx file

# already sorted correctly

# 2. Obtain key information about the grid
N = freq_mat.shape[0] # number of data points

minx = min(freq_mat[:,0])
miny = min(freq_mat[:,1])
minz = min(freq_mat[:,2]) # find the upper and lower corners of
    the grid
```

```
maxx = max(freq_mat[:,0])
maxy = max(freq_mat[:,1])
maxz = max(freq_mat[:,2])

widthx = maxx - minx # find the widths of the grid
widthy = maxy - miny
widthz = maxz - minz

xs = numpy.unique(freq_mat[:,0])
ys = numpy.unique(freq_mat[:,1])
zs = numpy.unique(freq_mat[:,2])

resx = xs[1]- xs[0]
resy = ys[1]- ys[0]
resz = zs[1]- zs[0]

#resx = freq_mat[(widthz+1)*(widthy+1),0] - freq_mat[0,0]
#resy = freq_mat[widthz+1,1] - freq_mat[0,1] # find the
    resolution of the grid
#resz = freq_mat[1,2] - freq_mat[0,2]

nx = (widthx) / resx + 1 # number of grid points in each
    dimension
ny = (widthy) / resy + 1 # need to add one because the
    subtraction leaves out an entire row
nz = (widthz) / resz + 1

# test to make sure all is well with the size of the grid and
    its dimensions
assert (nx * ny * nz) == N, "Something␣is␣wrong␣with␣the␣
    freq_mat␣array:␣it␣is␣not␣a␣prismatic␣shape"

# 3. write the header and footer
if parameters['SaveVolumetricDensityMap'] == True:
```

```
if parameters['CompressOutput'] == True: dx_file = gzip.open(parameters['OutputFilenamePrefix'] + "volumetric_density.dx.gz",'wb')
else: dx_file = open(parameters['OutputFilenamePrefix'] + "volumetric_density.dx",'w')

header = header_template % (nx, ny, nz, minx, miny, minz, resx, resy, resz, nx, ny, nz, N) # format the header
footer = footer_template # the footer needs no formatting
dx_file.write(header)
newline_counter = 1
for i in range(N): # write the data to the DX file
dx_file.write("%8.6e" % freq_mat[i,3])
if newline_counter == 3:
newline_counter = 0
dx_file.write("\n")
else:
dx_file.write("␣")
newline_counter += 1
dx_file.write(footer)
dx_file.close
return

class MultithreadingCalcVolumeTask(MultithreadingTaskGeneral):
'''A class for calculating the volume.'''

def value_func(self, item, results_queue):
"""Calculate the volume.

Arguments:
item -- A list or tuple, the input data required for the calculation
results_queue -- A multiprocessing.Queue() object for storing the calculation output

```

```
"""

frame_indx = item[0]
pdb = item[1]
parameters = item[2]
# pts = parameters['pts_orig'].copy() # this works
pts = parameters['pts_orig'] # also works, so keep because
    faster

# if the user wants to save empty points (points that are
    removed), then we need a copy of the original
if parameters['OutputEqualNumPointsPerFrame'] == True:
pts_deleted = pts.copy()

# you may need to load it from disk if the user so specified
if parameters['UseDiskNotMemory'] == True: # so you need to load
     it from disk
pym_filename = pdb
pdb = pymolecule.Molecule()
pdb.fileio.load_pym_into(pym_filename)

# remove the points that are far from the points region anyway
min_pts = numpy.min(pts,0) - parameters['DistanceCutoff'] - 1
max_pts = numpy.max(pts,0) + parameters['DistanceCutoff'] + 1

# identify atoms that are so far away from points that they can
    be ignored
index_to_keep1 = numpy.nonzero((pdb.information.coordinates[:,0]
     > min_pts[0]))[0] # x's too small
index_to_keep2 = numpy.nonzero((pdb.information.coordinates[:,0]
     < max_pts[0]))[0] # x's too large
index_to_keep3 = numpy.nonzero((pdb.information.coordinates[:,1]
     > min_pts[1]))[0] # y's too small
index_to_keep4 = numpy.nonzero((pdb.information.coordinates[:,1]
     < max_pts[1]))[0] # y's too large
```

```
index_to_keep5 = numpy.nonzero((pdb.information.coordinates[:,2]
     > min_pts[2]))[0] # z's too small
index_to_keep6 = numpy.nonzero((pdb.information.coordinates[:,2]
     < max_pts[2]))[0] # z's too large

index_to_keep = numpy.intersect1d(index_to_keep1, index_to_keep2
    , assume_unique=True)
index_to_keep = numpy.intersect1d(index_to_keep, index_to_keep3,
     assume_unique=True)
index_to_keep = numpy.intersect1d(index_to_keep, index_to_keep4,
     assume_unique=True)
index_to_keep = numpy.intersect1d(index_to_keep, index_to_keep5,
     assume_unique=True)
index_to_keep = numpy.intersect1d(index_to_keep, index_to_keep6,
     assume_unique=True)

# keep only relevant atoms
if len(index_to_keep) > 0: pdb = pdb.selections.
    create_molecule_from_selection(index_to_keep)

# get the vdw radii of each protein atom
vdw = numpy.ones(len(pdb.information.coordinates)) # so the
    default vdw is 1.0

# get vdw... you might want to fill this out with additional vdw
     values
vdw[numpy.nonzero(pdb.information.atom_information['
    element_stripped'] == "H")[0]] = 1.2
vdw[numpy.nonzero(pdb.information.atom_information['
    element_stripped'] == "C")[0]] = 1.7
vdw[numpy.nonzero(pdb.information.atom_information['
    element_stripped'] == "N")[0]] = 1.55
vdw[numpy.nonzero(pdb.information.atom_information['
    element_stripped'] == "O")[0]] = 1.52
```

```
vdw[numpy.nonzero(pdb.information.atom_information['
    element_stripped'] == "F")[0]] = 1.47
vdw[numpy.nonzero(pdb.information.atom_information['
    element_stripped'] == "P")[0]] = 1.8
vdw[numpy.nonzero(pdb.information.atom_information['
    element_stripped'] == "S")[0]] = 1.8
vdw = numpy.repeat(numpy.array([vdw]).T, len(pts), axis=1)

# now identify the points that are close to the protein atoms
dists = cdist(pdb.information.coordinates, pts)
close_pt_index = numpy.nonzero((dists < (vdw + parameters['
    DistanceCutoff'])))[1]

#save points inside protein or ligand
rememberedall = pts

# now keep the appropriate points
pts = numpy.delete(pts, close_pt_index, axis=0)

# exclude points outside convex hull
if parameters['ConvexHullExclusion'] == True:
convex_hull_3d = ConvexHull()

# get the coordinates of the non-hydrogen atoms (faster to
    discard hydrogens)
hydros = pdb.selections.select_atoms({'element_stripped':['H']})
not_hydros = pdb.selections.invert_selection(hydros)
not_hydros_coors = pdb.information.coordinates[not_hydros]

#not_hydros = pdb.selections.select_atoms({'name_stripped':['CA
    ']})
#not_hydros_coors = pdb.information.coordinates[not_hydros]

# modify pts here.
# note that the atoms of the pdb frame are in pdb.information.coordinates
#begintime = time.time() # measure execution time
akl_toussaint_pts = convex_hull_3d.akl_toussaint(not_hydros_coors) # quickly reduces input size
#print "akl Toussaint:", time.time() - begintime
begintime = time.time() # measure execution time
hull = convex_hull_3d.gift_wrapping_3d(akl_toussaint_pts) # calculate convex hull using gift wrapping algorithm
#print "gift_wrapping:", time.time() - begintime

old_pts = pts # we will need to regenerate the pts list, disregarding those outside the hull
pts = []
for pt in old_pts:
pt_outside = convex_hull_3d.outside_hull(pt, hull) # check if pt is outside hull
if not pt_outside:
pts.append(pt) # if its not outside the hull, then include it in the volume measurement
pts = numpy.array(pts)

# Now, enforce contiguity if needed
if len(parameters['ContiguousPocketSeedRegions']) > 0 and len(pts) > 0:
# first, for each point, determine how many neighbors it has
cutoff_dist = parameters['GridSpacing'] * 1.01 * math.sqrt(3) # to count kiddy-corner points too
pts_dists = squareform(pdist(pts))
neighbor_counts = numpy.sum(pts_dists < cutoff_dist,axis=0) - 1 # minus 1 because an atom shouldn't be considered its own neighor

# remove all the points that don't have enough neighbors
```

```
pts = pts[numpy.nonzero(neighbor_counts >= parameters['
    ContiguousPointsCriteria'])[0]]

# get all the points in the defined parameters['ContiguousPocket
    '] seed regions
contig_pts = parameters['ContiguousPocketSeedRegions'][0].
    points_set(parameters['GridSpacing'])
for Contig in parameters['ContiguousPocketSeedRegions'][1:]:
    contig_pts = numpy.vstack((contig_pts, Contig.points_set(
    parameters['GridSpacing'])))
contig_pts = unique_rows(contig_pts)

try: # error here if there are no points of contiguous seed
    region outside of protein volume.
# now just get the ones that are not near the protein
contig_pts = pts[numpy.nonzero(cdist(contig_pts, pts) < 1e-7)
    [1]]

last_size_of_contig_pts = 0
while last_size_of_contig_pts != len(contig_pts):
last_size_of_contig_pts = len(contig_pts)

# now get the indecies of all points that are close to the
    contig_pts
all_pts_close_to_contig_pts_boolean = (cdist(pts, contig_pts) <
    cutoff_dist)
index_all_pts_close_to_contig_pts = numpy.unique(numpy.nonzero(
    all_pts_close_to_contig_pts_boolean)[0])
contig_pts = pts[index_all_pts_close_to_contig_pts]

pts = contig_pts

except:
log("\tFrame␣" + str(frame_indx) + ":␣None␣of␣the␣points␣in␣the␣
    contiguous-pocket␣seed␣region\n\t\tare␣outside␣the␣volume␣of␣
    the␣protein!␣Assuming␣a␣pocket\n\t\tvolume␣of␣0.0␣A.",
    parameters)
pts = numpy.array([])

#make lists from whole set and final pts set
print (rememberedall)
rememberedall_list = rememberedall.tolist()
print (rememberedall_list)

print (pts)
pts_list = pts.tolist()
print (pts_list)

# find not-shared points of both lists
rest = tuple(x for x in rememberedall_list if x not in pts_list)
print (rest)
restarray = numpy.asarray(rest)

# write the remembered points (inside protein) to .pdb
if parameters['SaveIndividualPocketVolumes'] == True:
remem_text = ""
 remem_text = remem_text + "REMARK␣Frame␣" + str(frame_indx) + "
     \n"
 remem_text = remem_text + numpy_to_pdb(restarray, 'X')
 remem_text = remem_text + "END\n"

fl = open(parameters['OutputFilenamePrefix'] + "
    inLigOrProt_frame_" + str(frame_indx) + ".pdb", 'w')
fl.write(remem_text)
fl.close

# now write the pdb and calculate the volume
volume = len(pts) * math.pow(parameters['GridSpacing'],3)

```

```
log("\tFrame␣" + str(frame_indx) + ":␣" + repr(volume) + "␣A^3", parameters)
if parameters['SaveIndividualPocketVolumes'] == True:
frame_text = ""
frame_text = frame_text + "REMARK␣Frame␣" + str(frame_indx) + "\n"
frame_text = frame_text + "REMARK␣Volume␣=␣" + repr(volume) + "␣Cubic␣Angtroms\n"
frame_text = frame_text + numpy_to_pdb(pts, 'X')

if parameters['OutputEqualNumPointsPerFrame'] == True:
# you need to find the points that are in pts_deleted but not in pts
tmp = reduce(lambda x, y: x | numpy.all(pts_deleted == y, axis=-1), pts, numpy.zeros(pts_deleted.shape[:1], dtype=numpy.bool))
indices = numpy.where(tmp)[0]
pts_deleted = numpy.delete(pts_deleted, indices, axis=0)

pts_deleted = numpy.zeros(pts_deleted.shape) # So extra points will always be at the origin. These can be easily hidden with your visualization software.
frame_text = frame_text + numpy_to_pdb(pts_deleted, 'X',"XXX")

frame_text = frame_text + "END\n"

if parameters['CompressOutput'] == True: fl = gzip.open(parameters['OutputFilenamePrefix'] + "frame_" + str(frame_indx) + ".pdb.gz", 'wb')
else: fl = open(parameters['OutputFilenamePrefix'] + "frame_" + str(frame_indx) + ".pdb", 'w')
fl.write(frame_text)
fl.close()

extra_data_to_add = {}
```

```
if parameters['SaveVolumetricDensityMap'] == True:
    extra_data_to_add['SaveVolumetricDensityMap'] = pts

self.results.append((frame_indx, volume, extra_data_to_add))

#if len(extra_data_to_add.keys()) != 0:
#else: self.results.append((frame_indx, volume))

class MultithreadingStringToMoleculeTask(
    MultithreadingTaskGeneral):
'''A class for loading PDB frames (as strings) into pymolecule.
    Molecule objects.'''

def value_func(self, item, results_queue):
"""Convert a PDB string into a pymolecule.Molecule object

Arguments:
item -- A list or tuple, the input data required for the
    calculation
results_queue -- A multiprocessing.Queue() object for storing
    the calculation output

"""

pdb_string = item[0]
index = item[1]
parameters = item[2]

# make the pdb object
str_obj = StringIO(pdb_string)
tmp = pymolecule.Molecule()
tmp.fileio.load_pdb_into_using_file_object(str_obj, False, False
    , False)

log("\tFurther␣processing␣frame␣" + str(index), parameters)
```

```

if parameters['UseDiskNotMemory'] == False: # so load the whole
    trajectory into memory
self.results.append((index, tmp))
else: # save to disk, record filename
pym_filename = "." + os.sep + ".povme_tmp" + os.sep + "frame_" +
     str(index) + ".pym"
tmp.fileio.save_pym(pym_filename, False, False, False, False,
    False)
self.results.append((index, pym_filename))

class Region:
'''A class for defining regions that will be filled with points.
    '''

def __init__(self):
'''Initialize some variables.'''

self.center = numpy.array([9999.9, 9999.9, 9999.9])
self.radius = 9999.9 # in case the region is a sphere
self.box_dimen = numpy.array([9999.9, 9999.9, 9999.9]) # in case
     the region is a box

self.region_type = "SPHERE" # could also be BOX

def __str__(self):
'''Returns a string representation of the region.'''

if self.region_type == "SPHERE": return "sphere␣at␣(" + str(self
    .center[0]) + ",␣" + str(self.center[1]) + ",␣" + str(self.
    center[2]) + "),␣radius␣=␣" + str(self.radius)
if self.region_type == "BOX": return "box␣centered␣at␣(" + str(
    self.center[0]) + ",␣" + str(self.center[1]) + ",␣" + str(
    self.center[2]) + ")␣with␣x,y,z␣dimensions␣of␣(" + str(self.
    box_dimen[0]) + ",␣" + str(self.box_dimen[1]) + ",␣" + str(
    self.box_dimen[2]) + ")"
return ''

def __snap(self, pts, reso):
"""Snaps a set of points to a fixed grid.

Arguments:
pts -- A nx3 numpy.array representing 3D points.
reso -- A float, the resolution of the grid.

Returns:
A nx3 numpy.array with the 3D points snapped to the nearest grid
     point.

"""

# unfortunately, numpy.around rounds evenly, so 0.5 rounds to
    0.0 and 1.5 rounds to 2.0.
# very annoying, I'll just add a tiny amount to 0.5 => 0.500001
# this should work, since user is unlikely to select region
    center or radius with such
# precision

pts = pts + 1e-10
return numpy.around(pts/reso) * reso

def points_set(self, reso):
"""Generates a point field by filling the region with equally
    spaced points.

Arguments:
reso -- A float, the resolution of the grid on which the points
    will be placed.

Returns:
```

```
A nx3 numpy.array with the 3D points filling the region.

"""

total_pts = None

if self.region_type == "BOX":
xs = numpy.arange(self.center[0] - self.box_dimen[0] / 2, self.
    center[0] + self.box_dimen[0] / 2, reso)
ys = numpy.arange(self.center[1] - self.box_dimen[1] / 2, self.
    center[1] + self.box_dimen[1] / 2, reso)
zs = numpy.arange(self.center[2] - self.box_dimen[2] / 2, self.
    center[2] + self.box_dimen[2] / 2, reso)

total_pts = numpy.empty((len(xs) * len(ys) * len(zs), 3))

i = 0
for x in xs:
for y in ys:
for z in zs:
total_pts[i][0] = x
total_pts[i][1] = y
total_pts[i][2] = z

i = i + 1

total_pts = self.__snap(total_pts, reso)

elif self.region_type == "SPHERE":
xs = numpy.arange(self.center[0] - self.radius, self.center[0] +
     self.radius, reso)
ys = numpy.arange(self.center[1] - self.radius, self.center[1] +
     self.radius, reso)
zs = numpy.arange(self.center[2] - self.radius, self.center[2] +
     self.radius, reso)
```

```

total_pts = numpy.empty((len(xs) * len(ys) * len(zs), 3))

i = 0
for x in xs:
for y in ys:
for z in zs:
total_pts[i][0] = x
total_pts[i][1] = y
total_pts[i][2] = z

i = i + 1

total_pts = self.__snap(total_pts, reso)

# now remove all the points outside of this sphere
index_inside_sphere = numpy.nonzero(cdist(total_pts, numpy.array
    ([self.center])) < self.radius)[0]
total_pts = total_pts[index_inside_sphere]

return total_pts

class ConfigFile:
'''A class for processing the user-provided configuration file.
    '''

entities = []

def __init__ (self, FileName):
"""Generates a point field by filling the region with equally
    spaced points.

Arguments:
FileName -- A string, the filename of the configuration file.

```

```
"""

f = open(FileName, 'r')
lines = f.readlines()
f.close()

for line in lines:
# remove comments
line = line.split("#",1)[0]
# line = line.split("//",1)[0] # We can't have these kinds of comments any more because of Windows filenames.

line = line.strip()

if line != "":

# replace ; and , and : with space
# line = line.replace(',',' ')
# line = line.replace(';',' ')
# line = line.replace(':',' ') # this messes up Windows filenames
line = line.replace("\t",'␣')

# now strip string
line = line.strip()

# now, replace double spaces with one space
while '␣␣' in line: line = line.replace('␣␣','␣')

# Now split the thing
line = line.split('␣',1)

# now, make it upper case
line[0] = line[0].upper()

```

```
# If there's QUIT, EXIT, or STOP, then don't continue.
if line[0] in ['QUIT','EXIT','STOP']: break

self.entities.append(line)

class runit():
'''The main class to run POVME.'''

def reference(self, parameters, before=""):
'''Print out a message regarding terms of use.'''

log("", parameters)
log(before + "If␣you␣use␣POVME␣in␣your␣research,␣please␣cite␣the␣following␣reference:", parameters)
log(before + "␣␣Durrant,␣J.␣D.,␣C.␣A.␣de␣Oliveira,␣et␣al.␣(2011).␣\"POVME:␣An␣algorithm", parameters)
log(before + "␣␣for␣measuring␣binding-pocket␣volumes.\"␣J␣Mol␣Graph␣Model␣29(5):␣773-776.", parameters)

def load_multi_frame_pdb(self, filename, parameters):
"""Load a multi-frame PDB into memory or into separate files (depending on user specifications).

Arguments:
filename -- A string, the filename of the multi-frame PDB
parameters -- A python dictionary, where the keys are the user-defined parameter names and the values are the corresponding parameter values.

Returns:
If the user has requested that the disk be used to save memory, this function returns a list of tuples, where the first item in each tuple is the frame index, and the second is a filename containing the individual frame.
If memory is to be used instead of the disk, this function
    returns a list of tuples, where the first item in each tuple
    is the frame index, and the second is a pymolecule.Molecule
    object representing the frame.

"""

pdb_strings = []
growing_string = ''

log("", parameters)
log("Reading␣frames␣from␣" + filename, parameters)

f = open(filename, 'rb')
while True:

if parameters['NumFrames'] != -1:
if len(pdb_strings) >= parameters['NumFrames']: break

line = f.readline()

if len(line) == 0:
pdb_strings.append(growing_string)
break
if line[:3] == "END":
pdb_strings.append(growing_string)
growing_string = ''
else:
growing_string = growing_string + line

f.close()

while '' in pdb_strings: pdb_strings.remove('')

# now convert each pdb string into a pymolecule.Molecule object
```

```
molecules = Multithreading([(pdb_strings[idx], idx + 1,
    parameters) for idx in range(len(pdb_strings))], parameters['
    NumProcessors'], MultithreadingStringToMoleculeTask)
molecules = molecules.results

return molecules

def __init__(self, argv):
'''Start POVME

Arguments:
argv -- A list of the command-line arguments.

'''

start_time = time.time()

# Load the configuration file
if len(argv) == 1:
print "\nPOVME␣" + version
print "\nPlease␣specify␣the␣input␣file␣from␣the␣command␣line!\n\
    nExample:␣python␣POVME.py␣input_file.ini"
self.reference({})
print
sys.exit()

config = ConfigFile(argv[1])

# Process the config file
parameters = {}

parameters['GridSpacing'] = 1.0 # default
parameters['PointsIncludeRegions'] = []
parameters['PointsExcludeRegions'] = []
parameters['SavePoints'] = False # default
```

```
parameters['LoadPointsFilename'] = '' # default
parameters['PDBFileName'] = "" # default
parameters['DistanceCutoff'] = 1.09 # default is VDW radius of
    hydrogen
parameters['ConvexHullExclusion'] = True
parameters['ContiguousPocketSeedRegions'] = []
parameters['ContiguousPointsCriteria'] = 4
parameters['NumProcessors'] = 4
parameters['UseDiskNotMemory'] = False
parameters['OutputFilenamePrefix'] = "POVME_output." + time.
    strftime("%m-%d-%y") + "." + time.strftime("%H-%M-%S") + os.
    sep
parameters['SaveIndividualPocketVolumes'] = False
parameters['SavePocketVolumesTrajectory'] = False
parameters['OutputEqualNumPointsPerFrame'] = False
parameters['SaveTabbedVolumeFile'] = False
parameters['SaveVolumetricDensityMap'] = False
parameters['CompressOutput'] = False
parameters['NumFrames'] = -1 # This is a parameter for debugging
     purposes only.

float_parameters = ["GridSpacing", "DistanceCutoff"]
boolean_parameters = ["SavePoints", "ConvexHullExclusion", "
    CompressOutput", "UseDiskNotMemory", "
    SaveVolumetricDensityMap", "OutputEqualNumPointsPerFrame", "
    SaveIndividualPocketVolumes", "SaveTabbedVolumeFile", "
    SavePocketVolumesTrajectory"]
int_parameters = ["NumFrames", "ContiguousPointsCriteria", "
    NumProcessors"]
string_parameters = ["OutputFilenamePrefix", "PDBFileName", "
    LoadPointsFilename"]

print config.entities

for entity in config.entities:
```

```
try:
index = [p.upper() for p in float_parameters].index(entity[0])
parameters[float_parameters[index]] = float(entity[1])
except: pass

try:
index = [p.upper() for p in boolean_parameters].index(entity[0])
if entity[1].upper() in ["YES", "TRUE"]: parameters[
    boolean_parameters[index]] = True
else: parameters[boolean_parameters[index]] = False
except: pass

try:
index = [p.upper() for p in int_parameters].index(entity[0])
parameters[int_parameters[index]] = int(entity[1])
except: pass

try:
index = [p.upper() for p in string_parameters].index(entity[0])
parameters[string_parameters[index]] = entity[1].strip()
except: pass

# Regions are handled separately for each parameter...
if entity[0] == "POINTSINCLUSIONSPHERE":
Include = Region()
items = entity[1].split('␣')
Include.center[0] = float(items[0])
Include.center[1] = float(items[1])
Include.center[2] = float(items[2])
Include.radius = float(items[3])
Include.region_type = "SPHERE"
parameters['PointsIncludeRegions'].append(Include)
elif entity[0] == "POINTSINCLUSIONBOX":
Include = Region()
items = entity[1].split('␣')
```

```
Include.center[0] = float(items[0])
Include.center[1] = float(items[1])
Include.center[2] = float(items[2])
Include.box_dimen[0] = float(items[3])
Include.box_dimen[1] = float(items[4])
Include.box_dimen[2] = float(items[5])
Include.region_type = "BOX"
parameters['PointsIncludeRegions'].append(Include)
if entity[0] == "CONTIGUOUSPOCKETSEEDSPHERE":
Contig = Region()
items = entity[1].split('␣')
Contig.center[0] = float(items[0])
Contig.center[1] = float(items[1])
Contig.center[2] = float(items[2])
Contig.radius = float(items[3])
Contig.region_type = "SPHERE"
parameters['ContiguousPocketSeedRegions'].append(Contig)
elif entity[0] == "CONTIGUOUSPOCKETSEEDBOX":
Contig = Region()
items = entity[1].split('␣')
Contig.center[0] = float(items[0])
Contig.center[1] = float(items[1])
Contig.center[2] = float(items[2])
Contig.box_dimen[0] = float(items[3])
Contig.box_dimen[1] = float(items[4])
Contig.box_dimen[2] = float(items[5])
Contig.region_type = "BOX"
parameters['ContiguousPocketSeedRegions'].append(Contig)
elif entity[0] == "POINTSEXCLUSIONSPHERE":
Exclude = Region()
items = entity[1].split('␣')
Exclude.center[0] = float(items[0])
Exclude.center[1] = float(items[1])
Exclude.center[2] = float(items[2])
Exclude.radius = float(items[3])
Exclude.region_type = "SPHERE"
parameters['PointsExcludeRegions'].append(Exclude)
elif entity[0] == "POINTSEXCLUSIONBOX":
Exclude = Region()
items = entity[1].split('␣')
Exclude.center[0] = float(items[0])
Exclude.center[1] = float(items[1])
Exclude.center[2] = float(items[2])
Exclude.box_dimen[0] = float(items[3])
Exclude.box_dimen[1] = float(items[4])
Exclude.box_dimen[2] = float(items[5])
Exclude.region_type = "BOX"
parameters['PointsExcludeRegions'].append(Exclude)

# If the output prefix includes a directory, create that
    directory if necessary
if os.sep in parameters['OutputFilenamePrefix']:
output_dirname = os.path.dirname(parameters['
    OutputFilenamePrefix'])
#if os.path.exists(output_dirname): shutil.rmtree(output_dirname
    ) # So delete the directory if it already exists.
try: os.mkdir(output_dirname)
except: pass

# print out the header
self.reference(parameters, "")
log('', parameters)

# create temp swap directory if needed
if parameters['UseDiskNotMemory'] == True:
if os.path.exists('.' + os.sep + '.povme_tmp'): shutil.rmtree('.
    ' + os.sep + '.povme_tmp')
os.mkdir('.' + os.sep + '.povme_tmp')

# print out parameters
```

```
log("Parameters:", parameters)
for i in parameters.keys():

if i == 'NumFrames' and parameters['NumFrames'] == -1: continue
    # So only show this parameter if it's value is not the
    default.

if type(parameters[i]) is list:
for i2 in parameters[i]:
if i2 != "": log("\t" + str(i) + ":␣" + str(i2), parameters)
else:
if parameters[i] != "": log("\t" + str(i) + ":␣" + str(
    parameters[i]), parameters)

pts = None
if len(parameters['PointsIncludeRegions']) > 0: # so create the
    point file

log("\nGenerating␣the␣pocket-encompassing␣point␣field",
    parameters)

# get all the points of the inclusion regions
pts = parameters['PointsIncludeRegions'][0].points_set(
    parameters['GridSpacing'])
for Included in parameters['PointsIncludeRegions'][1:]: pts =
    numpy.vstack((pts, Included.points_set(parameters['
    GridSpacing'])))
pts = unique_rows(pts)

# get all the points of the exclusion regions
if len(parameters['PointsExcludeRegions']) > 0:
pts_exclusion = parameters['PointsExcludeRegions'][0].points_set
    (parameters['GridSpacing'])
for Excluded in parameters['PointsExcludeRegions'][1:]:
    pts_exclusion = numpy.vstack((pts_exclusion, Excluded.
    points_set(parameters['GridSpacing'])))
pts_exclusion = unique_rows(pts_exclusion)

# remove the exclusion points from the inclusion points
# I think there ought to be a set-based way of doing this,
# but I'm going to go for the pairwise comparison.
# consider rewriting later
index_to_remove = numpy.nonzero(cdist(pts, pts_exclusion) < 1e
    -7)[0]
pts = numpy.delete(pts, index_to_remove, axis=0)

# save the points as PDB
if parameters['SavePoints'] == True:

# First, save the point field itself

log("\nSaving␣the␣point␣field␣as␣a␣PDB␣and␣NPY␣file", parameters
    )

points_filename = parameters['OutputFilenamePrefix'] + "
    point_field.pdb"

if parameters['CompressOutput'] == True: afile = gzip.open(
    points_filename + ".gz", 'wb')
else: afile = open(points_filename,'w')

afile.write(numpy_to_pdb(pts, "X"))
afile.close()

# save the points as npy
numpy.save(points_filename + ".npy", pts)

log("\tPoint␣field␣saved␣to␣" + points_filename + "␣to␣permit␣
    visualization", parameters)
```

```
log("\tPoint␣field␣saved␣to␣" + points_filename + ".npy␣to␣
    optionally␣load␣for␣the␣volume␣calculation", parameters)
log("", parameters)

# Now, save the contiguous seed points as well, if specified.
if len(parameters['ContiguousPocketSeedRegions']) > 0:
# get all the contiguous points
contig_pts = parameters['ContiguousPocketSeedRegions'][0].
    points_set(parameters['GridSpacing'])
for Contig in parameters['ContiguousPocketSeedRegions'][1:]:
    contig_pts = numpy.vstack((contig_pts, Contig.points_set(
    parameters['GridSpacing'])))
contig_pts = unique_rows(contig_pts)

log("\nSaving␣the␣contiguous-pocket␣seed␣points␣as␣a␣PDB␣file",
    parameters)

points_filename = parameters['OutputFilenamePrefix'] + "
    contiguous_pocket_seed_points.pdb"

if parameters['CompressOutput'] == True: afile = gzip.open(
    points_filename + ".gz", 'wb')
else: afile = open(points_filename,'w')

afile.write(numpy_to_pdb(contig_pts, "X"))
afile.close()

log("\tContiguous-pocket␣seed␣points␣saved␣to␣" +
    points_filename + "␣to␣permit␣visualization", parameters)
log("", parameters)

if parameters['PDBFileName'] != '': # so there's a PDB point
    specified for calculating the volume.

# load the points in they aren't already present
```

```
if pts is None:
log("\nLoading␣the␣point−field␣NPY␣file ... ", parameters)
parameters['pts_orig'] = numpy.load(parameters['
    LoadPointsFilename'])
else: parameters['pts_orig'] = pts

# load the PDB frames
index_and_pdbs = self.load_multi_frame_pdb(parameters['
    PDBFileName'], parameters)

# calculate all the volumes
log("", parameters)
log("Calculating␣the␣pocket␣volume␣of␣each␣frame", parameters)
tmp = Multithreading([(index, pdb_object, parameters) for index,
     pdb_object in index_and_pdbs], parameters['NumProcessors'],
    MultithreadingCalcVolumeTask)

# delete the temp swap directory if necessary
if parameters['UseDiskNotMemory'] == True:
if os.path.exists('.' + os.sep + '.povme_tmp'): shutil.rmtree('.
    ' + os.sep + '.povme_tmp')

# display the results
results_dic = {}
for result in tmp.results: results_dic[result[0]] = result[1]
log("", parameters)
log("FRAME␣␣␣␣␣␣␣␣|␣VOLUME␣(A^3)", parameters)
log("−−−−−−−−−−−−−+−−−−−−−−−−−−−", parameters)
for i in sorted(results_dic.keys()): log(str(i).ljust(13) + "|␣"
     + str(results_dic[i]), parameters)

log("", parameters)
log("Execution␣time␣=␣" + str(time.time()−start_time) + "␣sec",
    parameters)
log("", parameters)
```

```

# if the user requested a separate volume file, save that as well
if parameters['SaveTabbedVolumeFile'] == True:
if parameters['CompressOutput'] == True: f = gzip.open(parameters['OutputFilenamePrefix'] + "volumes.tabbed.txt.gz", 'wb')
else: f = open(parameters['OutputFilenamePrefix'] + "volumes.tabbed.txt", 'w')

for i in sorted(results_dic.keys()): f.write(str(i) + "\t" + str(results_dic[i]) + "\n")
f.close()

# if the user wanted a single trajectory containing all the volumes, generate that here.
if parameters['SavePocketVolumesTrajectory'] == True:
if parameters['CompressOutput'] == True: traj_file = gzip.open(parameters['OutputFilenamePrefix'] + "volume_trajectory.pdb.gz", 'wb')
else: traj_file = open(parameters['OutputFilenamePrefix'] + "volume_trajectory.pdb", 'w')

for frame_index in range(1,len(results_dic.keys())+1):
if parameters['CompressOutput'] == True: frame_file = gzip.open(parameters['OutputFilenamePrefix'] + "frame_" + str(frame_index) + ".pdb.gz", 'rb')
else: frame_file = open(parameters['OutputFilenamePrefix'] + "frame_" + str(frame_index) + ".pdb", 'r')

traj_file.write(frame_file.read())
frame_file.close()

traj_file.close()

```

```
# if the user requested a volumetric density map, then generate
    it here
if parameters['SaveVolumetricDensityMap'] == True:
unique_points = {}

overall_min = numpy.ones(3) * 1e100
overall_max = numpy.ones(3) * -1e100

for result in tmp.results:
pts = result[2]['SaveVolumetricDensityMap']

if len(pts) > 0:
amin = numpy.min(pts, axis=0)
amax = numpy.max(pts, axis=0)

overall_min = numpy.min(numpy.vstack((overall_min, amin)), axis
    =0)
overall_max = numpy.max(numpy.vstack((overall_max, amax)), axis
    =0)

for pt in pts:
pt_key = str(pt[0]) + ";" + str(pt[1]) + ";" + str(pt[2])
try: unique_points[pt_key] = unique_points[pt_key] + 1
except: unique_points[pt_key] = 1
if overall_min[0] == 1e100:
log("ERROR!␣Cannont␣save␣volumetric␣density␣file␣because␣no␣
    volumes␣present␣in␣any␣frame.", parameters)
else:
xpts = numpy.arange(overall_min[0], overall_max[0] + parameters[
    'GridSpacing'], parameters['GridSpacing'])
ypts = numpy.arange(overall_min[1], overall_max[1] + parameters[
    'GridSpacing'], parameters['GridSpacing'])
zpts = numpy.arange(overall_min[2], overall_max[2] + parameters[
    'GridSpacing'], parameters['GridSpacing'])

```

```
all_pts = numpy.zeros((len(xpts)*len(ypts)*len(zpts), 4))

i = 0
for x in xpts:
for y in ypts:
for z in zpts:
key = str(x) + ";" + str(y) + ";" + str(z)
all_pts[i][0] = x
all_pts[i][1] = y
all_pts[i][2] = z

try: all_pts[i][3] = unique_points[key]
except: pass

i = i + 1

# convert the counts in the fourth column into frequencies
all_pts[:,3] = all_pts[:,3] / len(tmp.results)
dx_freq(all_pts, parameters) # save the dx file

#print "To turn into a DX file:"
#print all_pts
#import cPickle as pickle
#pickle.dump(all_pts, open('dill.pickle', 'w'))

if __name__ == "__main__": dorun = runit(sys.argv)
```

A.2. Shape Complementarity R Script

```
###whole trajectory aligned on first frame, ligand and protein
    coordinates extracted seperately
###POVME maps created as described in the Experimental Section
###for later pairing with other frame parameters, make sure
    ordering of the input data according to the MD timeline is
```

```
maintained



workingDirectory <- "/some/working/directory"
setwd(workingDirectory)

###specification of input files

ligandPDB <- "pdbfile_of_ligandconfs/over_MD/without_headline.pdb"
startingPoints <- "file_of_starting_point_map/as_created_by_POVME.pdb"
povmeOutputDir <- "directory/of/povme/output_files/"
filePatternProtein <- "pattern/according/to/povmeoutput"    #something like "trjproteinpart.+inLigOrProt.+\\.pdb" to specifiy the POVME output files of the points inside the protein
filePatternLiginner <- #something like "trjligandpart[0-9][0-9]inner059frame\\_[0-9]+\\.pdb" to specifiy the POVME output files of the points surrounding the ligand with distance to heavy atoms of at least 0.59 Angstrom
filePatternLigouter <-   #something like "trjligandpart[0-9][0-9]innerframe\\_[0-9]+\\.pdb" to specifiy the POVME output files of the points surrounding the ligand with distance to heavy atoms of at least 1.59 Angstrom

###load required packages

library(doSNOW)
library(R.utils)
library(gdata)
library(RANN)


###load the ordering of atoms for the ligand
```

```
ligatomfilelength <- countLines(ligandPDB)
ligatomlength <- (grep("END", readLines(ligandPDB))[[1]]-1)
numberofframes <- (ligatomfilelength)/(ligatomlength+1)
ligatomlist <- list()
for (i in 1:numberofframes){ligatomlist[[i]] <- read.fwf(
    ligandPDB, widths = c(-8,3,5,-14,8,8,8,-23,1), skip=((i-1)*(
    ligatomlength+1)), nrow = ligatomlength)}
saveRDS(ligatomlist,"ligatomlist.rds")

###load starting point map

filelength <- countLines(startingPoints)
pointMaplist <- read.fwf(startingPoints, widths = c
    (-30,8,8,8,-24), nrow = filelength)
pointMapmatrix <- do.call(cbind, pointMaplist)
saveRDS(pointMapmatrix,"pointMapmatrix.rds")

###load point maps of the protein in the binding area

filelistProt <- list.files(path = povmeOutputDir, pattern =
    filePatternProtein)
filepathlistProt <- paste(povmeOutputDir, filelistProt, sep="")

datalistProt <- list()
sizeProt <- list()
for (i in 1:length(filepathlistProt)){sizeProt[[i]] <-
    countLines(filepathlistProt[[i]])}

cluster <- makeCluster(3)
registerDoSNOW(cluster)

datalistProt <- foreach(x=1:length(filepathlistProt), .
    errorhandling="remove") %dopar% read.fwf(filepathlistProt[[x
    ]], skip=1, widths=c(-30,8,8,8,-24), nrow=sizeProt[[x]]-2)
```

```

stopCluster(cluster)

saveRDS(datalistProt, "datalistProt.rds")

datalistnot0Prot <- list()
for (i in 1:length(datalistProt)) {datalistnot0Prot[[i]] <-
    datalistProt[[i]][apply(datalistProt[[i]], MARGIN=1,function(
    x) !all(x==0)), ]}
saveRDS(datalistnot0Prot,"datalistnot0Prot.rds")


###load ligand surrounding point maps with at least 1.59
    Angstrom distance to the ligand

filelistInvlig <- list.files(path = povmeOutputDir, pattern =
    filePatternLigouter)
filepathlistInvlig <- paste(povmeOutputDir, filelistInvlig, sep=
    "")

datalistInvlig <- list()
sizeInvlig <- list()
for (i in 1:length(filepathlistInvlig)){sizeInvlig[[i]] <-
    countLines(filepathlistInvlig[[i]])}

cluster <- makeCluster(3)
registerDoSNOW(cluster)

datalistInvlig <- foreach(x=1:length(filepathlistInvlig), .
    errorhandling="remove") %dopar% read.fwf(filepathlistInvlig[[
    x]], skip=2, widths=c(-30,8,8,8,-24), nrow=sizeInvlig[[x]]-3)

stopCluster(cluster)

saveRDS(datalistInvlig, "datalistInvlig.rds")
```

```
datalistnot0Invlig <- list()
 for (i in 1:length(datalistInvlig)) {datalistnot0Invlig[[i]] <-
     datalistInvlig[[i]][apply(datalistInvlig[[i]], MARGIN=1,
    function(x) !all(x==0)), ]}
saveRDS(datalistnot0Invlig,"datalistnot0Invlig.rds")


###load ligand surrounding point maps with at least 0.59
   Angstrom distance to the ligand

filelistInvliginner <- list.files(path = povmeOutputDir, pattern
    = filePatternLiginner)
filepathlistInvliginner <- paste(povmeOutputDir,
   filelistInvliginner, sep="")

datalistInvliginner <- list()
sizeInvliginner <- list()
for (i in 1:length(filepathlistInvliginner)){sizeInvliginner[[i
   ]] <- countLines(filepathlistInvliginner[[i]])}

cluster <- makeCluster(3)
registerDoSNOW(cluster)

datalistInvliginner <- foreach(x=1:length(
   filepathlistInvliginner), .errorhandling="remove") %dopar%
   read.fwf(filepathlistInvliginner[[x]], skip=2, widths=c
   (-30,8,8,8,-24), nrow=sizeInvliginner[[x]]-3)

stopCluster(cluster)

saveRDS(datalistInvliginner, "datalistInvliginner.rds")

datalistnot0Invliginner <- list()
```

```
 for (i in 1:length(datalistInvliginner)) {
    datalistnot0Invliginner[[i]] <- datalistInvliginner[[i]][
    apply(datalistInvliginner[[i]], MARGIN=1,function(x) !all(x
    ==0)), ]}
saveRDS(datalistnot0Invliginner,"datalistnot0Invliginner.rds")

###from both ligand surrounding maps calculate the ligand
    surrounding surface map points (overlap)

notduplicatedPoints <- list()
datalistLigIDs <- list()
datalistLig <- list()
for (t in 1:length(datalistnot0Invliginner)){notduplicatedPoints
    [[t]] <- !duplicated(rbind (datalistnot0Invlig[[t]],
    datalistnot0Invliginner[[t]]))
                         mapStart <- dim(datalistnot0Invlig[[t]])
                            [1]+1
                         mapEnd <- dim(rbind(datalistnot0Invlig[[t
                            ]], datalistnot0Invliginner[[t]]))[1]
                         datalistLigIDs[[t]] <- notduplicatedPoints
                            [[t]][mapStart:mapEnd]
                         datalistLig[[t]] <- datalistnot0Invliginner
                            [[t]][datalistLigIDs[[t]],]}


saveRDS(datalistLig, "datalistLig.rds")

###assign ligand atoms to near parts of the ligand surrounding
    point map

ligatomlistnoHs <- list()

for (l in 1:length(ligatomlist)){ligatomlistnoHs[[l]] <-
    ligatomlist[[l]][!(ligatomlist[[l]]["V6"]=="H"),]}

```

```

nnresult<-list()

for (l in 1:length(ligatomlist)){
        nnresult[[l]] <- nn2(ligatomlistnoHs[[l]][,3:5],
           datalistLig[[l]][,1:3], k=1, treetype="kd")
}

resultlist <- list()

for (l in 1:length(ligatomlist)){
results <- list()
        for (n in 1:length(nnresult[[l]][[1]])){
                results[[n]] <- ligatomlistnoHs[[1]][nnresult[[l
                   ]][[1]][n],2]
        }
resultlist[[l]] <- unlist(results)
}

for (l in 1:length(ligatomlist)){
datalistLig[[l]]["atom"] <- resultlist[[l]]
}

saveRDS(datalistLig, "datalistLigAtom.rds")

###find overlap of ligand surrounding and protein surrounding
    maps

duplicatedPoints <- list()
overlapMapIDs <- list()
overlapMap <- list()
for (t in 1:length(datalistnot0Prot)){duplicatedPoints[[t]] <-
    duplicated(rbind (datalistnot0Prot[[t]], datalistLig[[t
    ]][1:3]))
```

```
                    mapStart <- dim(datalistnot0Prot[[t]])
                        [[1]]+1
                    mapEnd <- length(duplicatedPoints[[t]])
                    overlapMapIDs[[t]] <- duplicatedPoints[[t
                        ]][mapStart:mapEnd]
                    overlapMap[[t]] <- datalistLig[[t]][
                        overlapMapIDs[[t]],]}


saveRDS(overlapMap, "overlapMap.rds")

overlapLength <- list()
for (i in 1:length(overlapMap)){overlapLength[[i]] <- (dim(
    overlapMap[[i]])[[1]])}
saveRDS(overlapLength, "overlapLength.rds")

write.fwf((as.data.frame(unlist(overlapLength))), file="
    overlapLength.dat",rownames=FALSE, colnames=FALSE, quote=
    FALSE, justify="right", width=10)

overlapRatio <- list()
for (i in 1:length(overlapLength)){overlapRatio[[i]] <- (
    overlapLength[[i]]/(dim(datalistLig[[i]])[1]))}
saveRDS(overlapRatio, "overlapRatio.rds")

write.fwf((as.data.frame(unlist(overlapRatio))), file="
    overlapRatio.dat",rownames=FALSE, colnames=FALSE, quote=FALSE
    , justify="right", width=12)

ratiobyatom <- list()
for(l in 1:length(ligatomlist)){
ratiobyatom[[l]] <- table(overlapMap[[l]]$atom)/table(
    datalistLig[[l]]$atom)
}
saveRDS(ratiobyatom, "overlapRatiobyatom.rds")
```

```r


df <- lapply(ratiobyatom,as.data.frame)
tdf <- lapply(df,t)
tdf2 <- list()
for (l in 1:length(tdf)){
tdf2[[l]] <- as.numeric(tdf[[l]][2,])}
exp <- do.call(rbind,tdf2)
saveRDS(exp, "plotdata.rds")
names <- unlist(df[[1]][1])

write.fwf(exp, file="overlapRatiobyatom.dat",rownames=FALSE,
    colnames=FALSE, quote=FALSE, justify="right", width=12)
write.fwf(t(df[[1]][1]), file="colnamesOverlapRatiobyatom.dat",
    rownames=FALSE, colnames=FALSE, quote=FALSE, justify="right",
     width=12)
```

A.3. Cluster Step Selection R Script

```r
###whole trajectory aligned on first frame, only protein
    coordinates extracted
###POVME maps created as described in the Experimental Section
###for later pairing with other frame parameters, make sure
    ordering of the input data according to the MD timeline is
    maintained
###input is a datamatrix containing for each frame binary data
    on whether a grid point is present (part of the binding site)
     or not

workingDirectory <- "/some/working/directory"
setwd(workingDirectory)

###packages

library(ade4)
library(gdata)

###input data

matrix_bothprots_all_names <- readRDS("matrix_bothprots_all_
    names.rds")

###data from equilibration and bootstrapping

matrix_allfromeq <- matrix_bothprots_all_names[-c(1:3125,
    10420:13545, 20838:23963, 31256:34381, 41675:44800,
    52093:55218),]

saveRDS(matrix_allfromeq, "matrix_allfromeq.rds")

rm(matrix_bothprots_all_names)

set.seed(19)
sampledata<-matrix_allfromeq[sample(nrow(matrix_allfromeq),
    14598),]
saveRDS(sampledata, "sampledata_19.rds")
rm(matrix_allfromeq)

###distance matrix simple matching

matchingdist<-dist.binary(sampledata, method=2, diag=FALSE,
    upper=FALSE)

rm(sampledata)
###ward's clustering

wardsresults<-hclust(matchingdist, method="ward.D", )
saveRDS(wardsresults, "wardsresults_sample19fromeq.rds")

###analysis of levels 1:50 of clustering results
```

```
###get clusters

trees<-cutree(wardsresults, k=1:50)

treelist<-list()
for(i in 1:50)
{
        treelist[[i]]<-sort(trees[,i])
}

clusterlist<-list(list())
for(i in 1:50)
{
        cluster<-list()
        for(n in 1:i)
        {
                cluster[[n]]<-rownames(subset(as.data.frame(
                    treelist[[i]]), as.data.frame(treelist[[i]])
                    [,1]==n))
        }
clusterlist[[i]]<-cluster
}

saveRDS(clusterlist,"WardsClusters.rds")

clusterratiolist<-list(list())
clusterratioonlylist<-list(list())

###find out how many frames are in each cluster and which ratio
    of it belongs to PTP1B (named "1PTY....") frames

for(i in 1:50)
{
clusterratio<-list()
clusterratioonly<-list()
```

```
        for (n in 1:length(clusterlist[[i]]))
        {

                count<-0
                for (y in 1:length(clusterlist[[i]][[n]]))
                {
                if (grepl("1P",clusterlist[[i]][[n]][[y]])==TRUE)
                {count<-count+1} else{count<-count}
                }
                clusterratio[[n]]<-paste(count/length(clusterlist[[i]][[n]]),length(clusterlist[[i]][[n]]),sep="-")
                clusterratioonly[[n]]<- count/length(clusterlist[[i]][[n]])
        }
clusterratiolist[[i]]<-clusterratio
clusterratioonlylist[[i]]<-clusterratioonly
i<-i+1
}
saveRDS(clusterratiolist,"WardsClusterRatios.rds")
saveRDS(clusterratioonlylist,"WardsClusterRatiosOnly.rds")

###determine for each clustering step which clusters belong to 1B and which to TC (contain more than 92.5% or less than 7.5% PTP1B frames)

nbrofclustersteps<-length(clusterratioonlylist)

nbrs1Blist<-list()
nbrsTClist<-list()
for (step in 2:nbrofclustersteps)
{
        nbrs1Blist[[step]]<-which(clusterratioonlylist[[step]]>0.925)
        nbrsTClist[[step]]<-which(clusterratioonlylist[[step
            ]]<0.075)
}

###get names of frames for both the 1B and TC lists

names1Blist<-list(list())
for (step in 3:nbrofclustersteps)
{
        names1B<-list()
        for (l in 1: length(nbrs1Blist[[step]]))
        {
                names1B[[l]]<-clusterlist[[step]][[nbrs1Blist[[
                    step]][l]]]
        }

        names1Blist[[step]]<-names1B
}

namesTClist<-list(list())
for (step in 3:nbrofclustersteps)
{
        namesTC<-list()
        for (l in 1: length(nbrsTClist[[step]]))
        {
                namesTC[[l]]<-clusterlist[[step]][[nbrsTClist[[
                    step]][l]]]
        }

        namesTClist[[step]]<-namesTC
}


###clean up

```

```
rm(clusterratioonlylist)
rm(clusterlist)
rm(nbrs1Blist)
rm(nbrsTClist)

###get point map data corresponding to the framenames from
    original matrix

matrix_bothprots<-readRDS("../matrix_allfromeq.rds")


for (step in 3:nbrofclustersteps)
{
        data1B<-list()
        for (z in 1: length(names1Blist[[step]]))
        {
                data1B[[z]]<-matrix_bothprots[names1Blist[[step
                    ]][[z]],]
        }
        filename<-paste("data1B_step", step, ".rds", sep="")
        saveRDS(data1B, filename)
}

for (step in 3:nbrofclustersteps)
{
        dataTC<-list()
        for (z in 1: length(namesTClist[[step]]))
        {
                dataTC[[z]]<-matrix_bothprots[namesTClist[[step
                    ]][[z]],]
        }
        filename<-paste("dataTC_step", step, ".rds", sep="")
        saveRDS(dataTC, filename)
}

```

```

###calculate occupancy for each TC or 1B assigned cluster (how
    often is a point of the starting map part of the binding site
     throughout this cluster) for levels 3:50 (not much
    clustering has happened in level 1 and 2 therefore omitted)

for (step in 3:50)
{
filename<-paste("dataTC_step", step, ".rds", sep="")
dataTC<-readRDS(filename)
rm(filename)

        for (subcl in 1:length(dataTC))
        {
                name <- paste("MDSumsCl_", step, "_TCSubcl_",
                    subcl, sep="")
                subcluster<-assign(name, dataTC[[subcl]])
                MDSums<- colSums(subcluster)
                calculateratio<-function(MDSums, size=size<-dim(
                    subcluster)[1] )
                {ratio<-((MDSums/size)*100)
                return(ratio)}
                occupancy<-lapply(MDSums, calculateratio)
                occupancyround<-lapply(occupancy, round, digits
                    =2)
                occ<-unlist(occupancyround)
                filename<-paste("/home/alex/Desktop/
                    Rworkspacewards_allfromeq/seed19fromeq/
                    Occupancy_Cl_", step, "_TCSubcl_", subcl, ".
                    dat", sep="")
                write.fwf((as.data.frame(occ)),file=filename,
                    rownames=FALSE, colnames=FALSE, quote=FALSE,
                    justify="right", width=6)
                rdsname<-paste("Occupancy_Cl_", step, "_TCSubcl_
                    ", subcl, ".rds", sep="")
```

```
                        saveRDS(occ, rdsname)

                        rm(name)
                        rm(subcluster)
                        rm(MDSums)
                        rm(calculateratio)
                        rm(occupancy)
                        rm(occupancyround)
                        rm(occ)
                        rm(filename)
                        rm(rdsname)
            }

rm(dataTC)

}

for (step in 3:50)
{

filename<-paste("data1B_step", step, ".rds", sep="")
data1B<-readRDS(filename)
rm(filename)

            for (subcl in 1:length(data1B))
            {
                        name <- paste("MDSumsCl_", step, "_1BSubcl_",
                            subcl, sep="")
                        subcluster<-assign(name, data1B[[subcl]])
                        MDSums<- colSums(subcluster)
                        calculateratio<-function(MDSums, size=size<-dim(
                            subcluster)[1] )
                        {ratio<-((MDSums/size)*100)
                        return(ratio)}
                        occupancy<-lapply(MDSums, calculateratio)
```

```
                    occupancyround<-lapply(occupancy, round, digits
                        =2)
                    occ<-unlist(occupancyround)
                    filename<-paste("/home/alex/Desktop/
                        Rworkspacewards_allfromeq/seed19fromeq/
                        Occupancy_Cl_", step, "_1BSubcl_", subcl, ".
                        dat", sep="")
                    write.fwf((as.data.frame(occ)),file=filename,
                        rownames=FALSE, colnames=FALSE, quote=FALSE,
                        justify="right", width=6)
                    rdsname<-paste("Occupancy_Cl_", step, "_1BSubcl_
                        ", subcl, ".rds", sep="")
                    saveRDS(occ, rdsname)

                    rm(name)
                    rm(subcluster)
                    rm(MDSums)
                    rm(calculateratio)
                    rm(occupancy)
                    rm(occupancyround)
                    rm(occ)
                    rm(filename)
                    rm(rdsname)
            }

rm(data1B)
}

occupancyvar1Blist<-list()

###for the same levels investigate all 1B and TC assigned
    clusters: count how many points are present in more than 95%
    or less than 5% of the contained frames

for (step in 3:50)
```

```
{
cat("step",step)
occupancyvars1B<-list()
filename<-paste("/home/alex/Desktop/Rworkspacewards_allfromeq/
    seed19fromeq/data1B_step", step, ".rds", sep="")
data1B<-readRDS(filename)
subcl_length<-length(data1B)
rm(data1B)

        for (subcl in 1:subcl_length)
        {
        cat("subcl",subcl)
        rdsname<-paste("/home/alex/Desktop/Rworkspacewards_
            allfromeq/seed19fromeq/Occupancy_Cl_", step, "_1
            BSubcl_", subcl, ".rds", sep="")
        occupancy<-readRDS(rdsname)

        l<-1
        nbr<-0
        while (l < (length(occupancy)+1))
                {
                cat("step",step,"subcl", subcl,"l", l)
                if (occupancy[[l]]>95 || occupancy[[l]]<5){nbr<-
                    nbr+1
                cat("nbr",nbr)}
                l<-l+1
                }

        occupancyvars1B[[subcl]]<-nbr
        }

occupancyvar1Blist[[step]]<-occupancyvars1B
savename<-paste("OccVar1B_Cl_", step,  ".rds", sep="")
saveRDS(occupancyvars1B, savename)
}
```

```

saveRDS(occupancyvar1Blist, "occupancyvar1Blist.rds")

rm(list=ls(all=TRUE))

occupancyvarTClist<-list()

for (step in 3:50)
{

occupancyvarsTC<-list()
filename<-paste("/home/alex/Desktop/Rworkspacewards_allfromeq/
    seed19fromeq/dataTC_step", step, ".rds", sep="")
dataTC<-readRDS(filename)
subcl_length<-length(dataTC)
rm(dataTC)

        for (subcl in 1:subcl_length)
        {
        rdsname<-paste("Occupancy_Cl_", step, "_TCSubcl_", subcl
            , ".rds", sep="")
        occupancy<-readRDS(rdsname)

        l<-1
        nbr<-0
                while (l < (length(occupancy)+1))
                        {
                        if (occupancy[[l]]>95 || occupancy[[l
                            ]]<5){nbr<-nbr+1}
                        l<-l+1
                        }

        occupancyvarsTC[[subcl]]<-nbr
        }

```

```r
occupancyvarTClist[[step]]<-occupancyvarsTC
savename<-paste("OccVarTC_Cl_", step,  ".rds", sep="")
saveRDS(occupancyvarsTC, savename)
}

saveRDS(occupancyvarTClist, "occupancyvarTClist.rds")



x<-c()
y<-c()

for (step in 3:length(occupancyvar1Blist))
{
        for (scl in 1:length(occupancyvar1Blist[[step]]))
        {
        x<-append(x, step)
        y<-append(y, (occupancyvar1Blist[[step]][[scl]]/10427))
        }
}



for (step in 3:length(occupancyvarTClist))
{
        for (scl in 1:length(occupancyvarTClist[[step]]))
        {
        x<-append(x, step)
        y<-append(y, (occupancyvarTClist[[step]][[scl]]/10427))
        }
}

saveRDS(x, "clusterdecision_x_n.rds")
saveRDS(y, "clusterdecision_y_n.rds")

```

```


table<-cbind(x,y)
sortedtable<-table[order(x),]
d_f<-as.data.frame(sortedtable)

ymean<-c()
xstep<-c()
ysd<-c()

for (step in 3:50)
{
ymean_n<-mean(d_f[which(d_f$x==step),]$y)
ymean<-append(ymean, ymean_n)
ysd_n<-sd(d_f[which(d_f$x==step),]$y)
ysd<-append(ysd, ysd_n)
xstep<-append(xstep,step)
}

d_f_mean<-data.frame(xstep, ymean)

library(ggplot2)
step_plot<-ggplot() + theme_bw() + geom_point(aes(y = y, x = x),
    data = d_f, stat="identity", size=0.1) +geom_line(aes(y=ymean
    , x=xstep), data = d_f_mean, stat="identity", colour = "#
    fc0cc1", size=1) + geom_ribbon(aes(ymin = ymean - ysd[xstep
    -2], ymax = ymean + ysd[xstep-2], x=xstep), data = d_f_mean,
    fill = "#fc0cc1", alpha=0.2) + labs(x="clustering␣step", y="
    intra-cluster␣uniformness")+ theme(text = element_text(size =
     12))+ ylim(0.7, 0.87)+ xlim(0, 50) + scale_x_continuous(
    breaks=seq(0,50,2))

postscript("step_plot19.eps",width=10,height=4.5)
print(step_plot)
dev.off()
```

```
pdf("step_plot19.pdf",width=10,height=4.5)
print(step_plot)
dev.off()
```

A.4. Pose Consistency R Script

```
#
    ######################################################################

#script for calculating and extracting consistent pose
    representatives
#from protein-ligand docking output
#
    ######################################################################

#this version works on a splitted input file with all docking
    poses for a ligand
#in one file (for example with "csplit"; output file prefix "
    part")
#
    ######################################################################

#working directry and number of poses per ligand have to be
    specified below (see -->)
#
    ######################################################################



#-->specify folder were input files lie
setwd("/.../.../...")

#load packages
library(readr)
library(bio3d)
library(dbscan)
library(gdata)

#—>specify number of poses per ligand (has to be constant)
posesperlig<-25

#collect input files
filelist <- list.files(pattern="part")
moleculenumber<-length(filelist)/posesperlig

tokeep<-c()

for (m in 1:moleculenumber)
{
        #read input files
        start<-((m-1)*posesperlig)+1
        end<-m*25
        filename<-filelist[[start]]
        con<-file(filename, open="r")
        line<-readLines(con)
        atomnumber<-as.numeric(substr(line[[4]],0,3))
        close(con)
        rm(line)

        datalist<-list()
        count<-1
        for (f in start:end)
        {
                datalist[[count]]<-read_fwf(filelist[[f]], fwf_
                   widths(c(10,10,10,3,1)), skip=4, n_max =
                   atomnumber)
                count<-count+1
        }

        #convert heavy atom coordinates into matrix
        coordinate_list<-list()
        coordinates<-list()
        for (s in 1:posesperlig)

        {
                withoutHs<-datalist[[s]][!(datalist[[s]]["X4"]
                   == "H"),]
                coordinate_list[[s]]<-withoutHs[c("X1","X2","X3"
                   )]
                coordinates[[s]]<-c(do.call(rbind,coordinate_
                   list[[s]]))

        }

        matrix<-do.call(rbind, coordinates)

        #create rmsd matrix from all pose coordinates of each
           ligand
        rmsdmatrix<-rmsd(matrix, ncore=3)
        rmsddistmatrix<-as.dist(rmsdmatrix)



        #parameters for clustering can be changed here
        results<-dbscan(rmsddistmatrix, 1.5, minPts=8)

        index<-seq(1:25)
        table<-as.data.frame(cbind(index, results$cluster))

        #assign value 9 to all values in the lines(poses) that
           do not belong to a cluster (for each cluster number
           seperately)
        reducedmatrix<-list()
        if (max(results$cluster)>0)
        {
                for (r in 1:max(results$cluster))
                {
                        reducedmatrix[[r]]<-rmsdmatrix
                        reducedmatrix[[r]][!(results$cluster==r)
                            ,]<-9

                }
        }



        #for each row of a reduced matrix calculate the pose,
           which is the one with the lowest rmsd sum (similar to
            medioid pose)
        repres<-list()
        if (max(results$cluster)>0)
        {for (r in 1:length(reducedmatrix))
        {
        repres[[r]]<-which(apply(reducedmatrix[[r]], 1, sum)==
           min(apply(reducedmatrix[[r]], 1, sum)))
        }
        }

        zeros<-rep(0, posesperlig)

        if (max(results$cluster)>0)
                {
                        repres<-unlist(repres)
                        zeros[repres]<-1
                }

        tokeep<-c(tokeep, zeros)

}

```

```
#save all consistent pose representatives to file (can be merged
    with .sdf of docking poses for exmaple with MOE)
saveRDS(tokeep,"tokeep.rds")

index<-seq(1:length(tokeep))
table<-as.data.frame(cbind(index, tokeep))

write.fwf(table, file="tokeeptable.txt", rownames=FALSE,
   colnames=FALSE, quote=FALSE, justify="right", width=c(10,5))
```

B. Publications

Peer-reviewed Articles

[1] Maccari, R., Ettari, R., Adornato, I., Naß, A., Wolber, G., Bitto, A., Mannino, F., Aliquò, F., Bruno, G., Nicolò, F., Previti, S., Grasso, S., Zappalà, M., Ottanà, R. Identification of 2-thioxoimidazolidin-4-one derivatives as novel noncovalent proteasome and immunoproteasome inhibitors, *Bioorganic & Medicinal Chemistry Letters*, **28**(3), 278-283, **2018**.

https://doi.org/10.1016/j.bmcl.2017.12.053

[2] Liu, J., Chen, L., Joseph, J.F., Naß, A., Stoll, A., de la Torre, X., Botrè, F., Wolber, G., Parr, M.K., Bureik, M. Combined chemical and biotechnological production of 20βOH-NorDHCMT, a long-term metabolite of Oral-Turinabol (DHCMT), *Journal of Inorganic Biochemistry*, Epub ahead of print, **2018**.

https://doi.org/10.1016/j.jinorgbio.2018.02.020

[3] Ottanà, R., Paoli, P., Naß, A., Lori, G., Cardile, V., Adornato, I., Rotondo, A., Graziano, A.C.E., Wolber, G., Maccari, R. Discovery of 4-[(5-arylidene-4-oxothiazolidin-3-yl)-methyl]benzoic acid derivatives active as novel potent allosteric inhibitors of protein tyrosine phosphatase 1B: In silico studies and in vitro evaluation as and anti-inflammatory agents, *European Journal of Medicinal Chemistry*, **127**, 840-858, **2017**.

https://doi.org/10.1016/j.ejmech.2016.10.063

[4] Schaller, D., Hagenow, S., Alpert, G., Naß, A., Schulz, R., Bermudez, M., Stark, H., Wolber, G. Systematic Data Mining Reveals Synergistic H3R/MCHR1 Ligands, *ACS Medicinal Chemistry Letters*, **8**(6), 648-653, **2017**.

https://doi.org/10.1021/acsmedchemlett.7b00118

[5] Parr, M.K., Botrè, F., Naß, A., Hengevoss, J., Diel, P., Wolber, G. Ecdysteroids: A novel class of anabolic agents? *Biology of Sport*, **32**(2), 169–173, **2015**. https://doi.org/10.5604/20831862.1144420

Conferences

[1] Naß, A., & Wolber, G. Interaction Pattern Analysis – What are we Missing? Talk at the *11th International Conference on Chemical Structures*, Mai 27-31, **2018**, Noordwijkerhout, the Netherlands.

[2] Naß, A., & Wolber, G. Chemoinformatic Approaches towards Selective PTP1B Inhibitors. Talk at the *20. Frühjahrssymposium des GDCh Jungchemikerforums*, March 21-24, **2018**, Konstanz, Germany.

[3] Naß, A., & Wolber, G. New Methods to Assess Flexible Aspects of Protein-Ligand Shape Complementarity and their Application to Increase Selectivity of PTP1B Inhibitors. Poster presentation at the *13th German Conference on Chemoinformatics*, November 5-7, **2017**, Fulda, Germany.

[4] Naß, A., & Wolber, G. PTP1B Selectivity – Exploring the world of protein ligand interactions beyond hydrogen bond counts in static structures. Talk at the *EUROPIN Summer School on Drug Design/EUROPIN Retreat 2017*, September 17-22, **2017**, Vienna, Austria.

[5] Naß, A., & Wolber, G. Binding Site Shape Clustering and Shape Complementarity to Increase Selectivity of PTP1B Inhibitors. Talk at the *12th German Conference on Chemoinformatics*, November 6-8, **2016**, Fulda, Germany.

[6] Naß, A., & Wolber, G. Binding Site Shape Clustering to Modulate Selectivity Profiles of Protein Tyrosine Phosphatase Inhibitors. Talk at the *Young Scientists in Cancer Research Symposium*, May 26, **2016**, Berlin, Germany.

[7] Naß, A., & Wolber, G. A New Method for Binding Site Shape Clustering to Increase Selectivity of PTP1B Ligands. Poster presentation at the *11th German Conference on Chemoinformatics*, November 8-10, **2015**, Fulda, Germany.

[8] Naß, A., & Wolber, G. Protein Tyrosine Phaosphatase 1B - Combining Inhibitory Activity with Selectivity. Talk at the *Vienna Summer School Drug Design/EUROPIN Retreat 2015*, September 20-25, **2015**, Vienna, Austria.

[9] Naß, A., & Wolber, G. Fragment-Based Discovery of Key Parameters for PTP1B Selectivity. Poster presentation at the *Fragments 2015: 5th RSC-BMCS Fragment-based Drug Discovery*, March 22-24, **2015**, Cambridge, United Kingdom.

[10] Naß, A., & Wolber, G. Protein Tyrosine Phaosphatase 1B - Combining Inhibitory Activity with Selectivity. Talk at the *EUROPIN Retreat Gdansk 2014*, September 26, **2014**, Gdansk, Poland.

[11] Naß, A., & Wolber, G. Active and Selective: Fragment Based Design for PTP1B Inhibitors. Talk at *Der wissenschaftliche Nachwuchs stellt sich vor*, July 4, **2014**, Berlin, Germany.

[12] Naß, A., & Wolber, G. PTP1B – Combining Inhibitory Activity with Selectivity. Poster presentation at the *10th International Conference on Chemical Structures*, June 1-5, **2014**, Noordwijkerhout, the Netherlands.