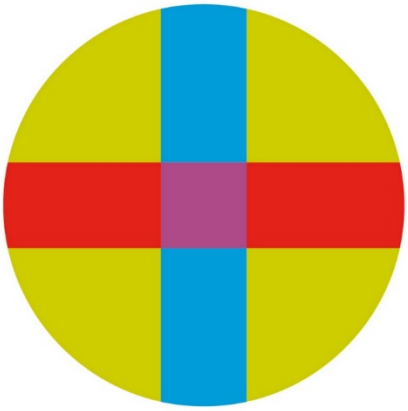
UNIVERSITY CEU - SAN PABLO

POLYTECHNIC SCHOOL

BIOMEDICAL ENGINEERING DEGREE



BACHELOR THESIS

# Development of a Workflow for Virtual Drug Screening using the Scipion Framework

Author: Verónica Gamo Parejo
Supervisors: Carlos Óscar Sorzano Sánchez and Javier Tejedor Noguerales

June 2025

UNIVERSIDAD SAN PABLO-CEU
ESCUELA POLITÉCNICA SUPERIOR
División de Ingeniería

## Datos del alumno

| NOMBRE: |
|---|
| |

## Datos del Trabajo

| TÍTULO DEL PROYECTO: |
|---|
| |

## Tribunal calificador

| PRESIDENTE: | FDO.: |
|---|---|
| | |

| SECRETARIO: | FDO.: |
|---|---|
| | |

| VOCAL: | FDO.: |
|---|---|
| | |

Reunido este tribunal el _____/_____/_____, acuerda otorgar al Trabajo Fin de Grado presentado por Don_____la calificación de _____.

# ACKNOWLEDGMENTS

Este proyecto marca el final de mi etapa universitaria, un período en el que he crecido tanto a nivel académico como personal. A lo largo de estos años, he comprendido la importancia del esfuerzo, la dedicación y la perseverancia para alcanzar mis objetivos. Además, he aprendido que el camino se hace más fácil cuando estás rodeado de personas que te apoyan y te inspiran a seguir adelante.

Quiero expresar mi más profundo agradecimiento a Carlos Óscar Sorzano, por confiar en mí desde el primer momento, valorarme tanto y darme la oportunidad de trabajar con el maravilloso grupo de Bioinformática del CNB. Su apoyo, paciencia y guía han sido fundamentales para mi crecimiento en este proyecto y su confianza en mis capacidades ha sido un gran impulso para seguir avanzando.

A mis amigas Sabela, Laura, Paula y Andrea, porque sin ellas este camino habría sido mucho más difícil. Gracias por ayudarme siempre, por estar ahí en los momentos de estrés y por hacer que estos años fueran más llevaderos con vuestra compañía y amistad incondicional.

A mi familia, por su amor incondicional, su apoyo constante y por creer en mí incluso en los momentos en los que yo misma dudaba. Gracias por ser mi refugio y mi mayor motivación para seguir adelante.

A esta Universidad, que tanto me ha enseñado y en la que he vivido experiencias inolvidables, que han contribuido a mi formación no solo como profesional, sino también como persona.

Finalmente, quiero agradecer a todas aquellas personas que, de una manera u otra, han formado parte de esta etapa y han contribuido a que este proyecto sea una realidad.

# ABSTRACT

This project presents the development of an automated and modular virtual drug screening (VDS) workflow within the Scipion-chem framework. The pipeline integrates ligand and protein preparation, binding site prediction, molecular docking with multiple engines, consensus analysis, rescoring using the Vina function, molecular dynamics (MD) simulations and machine learning–based affinity prediction. The workflow was validated using two therapeutically relevant protein targets, ROS1 and RIPK1, and a set of known inhibitors with experimental binding data, alongside drugs approved by the Food and Drug Administration (FDA) from the ZINC database. Among the docking engines tested, AutoDock-GPU demonstrated the most consistent and thermodynamically favourable performance, while AutoDock showed poor reliability. Consensus strategies based on binding energy and Vina scores improved the prioritization of ligands and partially reproduced experimental affinity rankings. MD simulations confirmed the structural stability of strong binders. Machine learning models (Pafnucy and PLAPT) offered additional predictive insights, but showed limited consistency, particularly in ranking high-affinity ligands. Overall, the workflow proved effective in filtering, ranking and validating candidate compounds, highlighting the importance of combining diverse computational methods. The modular architecture of Scipion-chem enables flexibility and reproducibility, offering a robust platform for early-stage drug discovery and a foundation for future enhancements, such as, absorption, distribution, metabolism, excretion and toxicity (ADMET) filtering and large-scale screening scalability.

# RESUMEN

Este proyecto presenta el desarrollo de un flujo de trabajo automatizado y modular para el cribado virtual de fármacos (VDS) dentro del entorno Scipion-chem. El flujo integra la preparación de ligandos y proteínas, predicción de sitios de unión, acoplamiento molecular con múltiples motores, análisis por consenso, valoración con la función de Vina, simulaciones de dinámica molecular (MD) y predicción de afinidad mediante modelos de aprendizaje automático. El flujo fue validado con dos dianas terapéuticas, ROS1 y RIPK1, utilizando inhibidores con datos experimentales y una biblioteca de compuestos aprobados por la Administración de Alimentos y Medicamentos (FDA) obtenida de la base de datos ZINC. AutoDock-GPU mostró el rendimiento más estable y termodinámicamente favorable, mientras que AutoDock resultó poco fiable. Las estrategias de consenso basadas en energía de unión y Vina permitieron una priorización más precisa de ligandos, reproduciendo parcialmente los rankings experimentales. Las simulaciones de MD confirmaron la estabilidad estructural de los ligandos de mayor afinidad. Los modelos Pafnucy y PLAPT ofrecieron predicciones complementarias, aunque con limitaciones en la clasificación de ligandos potentes. En conjunto, el flujo demostró ser eficaz en la priorización de candidatos, destacando el valor de combinar enfoques computacionales. Scipion-chem ofrece una plataforma flexible y reproducible, con potencial de expansión hacia filtrado ADMET (absorción, distribución, metabolismo, excreción y toxicidad) y cribado a gran escala.

# INDEX

# FIGURE INDEX

# TABLE INDEX

# EQUATION INDEX

# FIGURE INDEX - APPENDIX

# TABLE INDEX - APPENDIX

# 1 INTRODUCTION

Drug discovery is a challenging, time-consuming and costly process that involves identifying small molecules capable of modulating biological targets. It includes several stages [1], beginning with discovery and development, where potential compounds are studied for their biological effects, followed by preclinical research, where their safety, toxicity and pharmacokinetics are assessed through laboratory and animal testing. Promising candidates then move into clinical research, where they are tested in human volunteers across multiple phases to evaluate safety, efficacy and dosage. Once sufficient data is gathered, the Food and Drug Administration (FDA) reviews the findings to decide on approval or necessary modifications. Even after approval, the FDA continues monitoring through post-market surveillance to track long-term effects or adverse reactions. The process is lengthy and costly—developing a new drug is estimated to cost around $2.6 billion [2] and the chemical space of potential drug-like molecules is estimated at $10^{63}$ compounds [3], making the search extremely difficult.

To address these challenges, Virtual Drug Screening (VDS) has emerged as a computational approach that allows researchers to screen large chemical libraries and predict ligand-target interactions, significantly reducing the number of compounds requiring experimental testing and accelerating early-stage drug discovery.

## *1.1 State of the Art in Virtual Drug Screening*

As previously mentioned, VDS consists of computational techniques designed to identify promising drug candidates by filtering out compounds unlikely to interact with a biological target. Initial VDS approaches emerged in the 1990s with molecular docking methods like AutoDock [4], enabling the prediction of ligand-receptor interactions. With increased computational power and access to large chemical databases in the early 2000s [5], docking algorithms became more efficient and broadly applicable. A major advance came with the release of AutoDock Vina in 2010 [6], which significantly improved docking speed and accuracy. Subsequent developments, such as high-throughput virtual screening, machine learning and molecular dynamics (MD) simulations [7], have further enhanced VDS by offering more dynamic and detailed evaluations of ligand-protein

interactions. MD simulations help overcome the limitations of traditional rigid docking models by analysing ligand-target stability over time.

VDS methods are mainly classified into Structure-Based Virtual Screening (SBVS) and Ligand-Based Virtual Screening (LBVS) [4]. SBVS uses the receptor's physicochemical properties, often a protein, to identify ligands. It includes molecular docking, which predicts ligand binding orientation and estimates binding affinity using scoring functions, with tools such as AutoDock, Vina and Schrödinger Glide [8]. Another SBVS technique is the novo drug design [4], which generates new molecules designed for specific targets. In contrast, LBVS relies on the structural features of known active compounds to identify new candidates. Techniques like Quantitative Structure-Activity Relationship (QSAR) modelling [9] establish statistical correlations between molecular descriptors and biological activity. Pharmacophore modelling [10] also plays a key role by identifying the essential chemical features required for activity, guiding the search for structurally similar bioactive molecules.

To support these approaches, various computational tools have been developed, including LePhar, Fpocket and P2Rank [8], along with the aforementioned Schrödinger and AutoDock. However, many of these tools are tailored for specific tasks, often requiring researchers to integrate multiple platforms to perform a complete analysis. More recently, innovations such as artificial intelligence, cloud computing and multi-scale modelling [11] have further transformed VDS. These advancements have improved its efficiency, accuracy and predictive capabilities, making it an increasingly powerful asset in modern drug discovery.

## *1.2 Project Overview and Objectives*

This project is part of a larger initiative: Scipion-chem, a framework designed to extend Scipion's capabilities for computational chemistry and drug discovery. Scipion is a workflow engine that is particularly well suited for structural studies of biological macromolecules [12]. Although it was originally created in Madrid, its development has become a collaborative effort involving several academic institutions, with the Centro Nacional de Biotecnología (CNB-CSIC) remaining one of its principal contributors [12].

During my internship at CNB, I gained hands-on experience with Scipion, learning about its functionalities and practical applications. For this Final Degree Project, I have applied the knowledge acquired during that internship, alongside ongoing research efforts at CNB, to develop a workflow that automates SBVS within Scipion.

The main objective of this project is to design and implement a structured and automated SBVS workflow that integrates molecular docking, scoring and MD simulations into a single, reproducible pipeline. This approach addresses a key limitation of existing drug discovery software, where most tools specialize in individual tasks, but do not offer a fully integrated workflow. To achieve this goal, the project pursues the following specific objectives:

- Develop a modular and automated SBVS workflow within the Scipion-chem environment.
- Integrate the main stages of virtual screening into a single pipeline.
- Improve reproducibility and minimize manual intervention throughout the screening process.
- Validate the workflow using two therapeutically relevant protein targets: Proto-oncogene tyrosine-protein kinase 1 (ROS1) and Receptor-interacting serine/threonine-protein kinase 1 (RIPK1).
- Assess its performance in terms of accuracy, reproducibility and scalability.
- Contribute to the broader Scipion-chem initiative by adding tools and functionalities useful for computational drug discovery.

By consolidating these processes within Scipion-chem, this project aims to provide a standardized, efficient and user-friendly approach to SBVS, enhancing both reproducibility and accessibility for future research.

# 2 MATERIAL AND METHODS

This section outlines the materials and methods used in the study, emphasizing the use of Scipion and Scipion-chem as core computational tools. It details the design and implementation of an SBVS workflow and the integration of new plugins and custom protocols to extend Scipion-chem's functionality.

## *2.1 The Scipion Platform*

As introduced in Section 1.2, Scipion is a workflow engine designed to generate 3D models of macromolecular complexes from electron microscopy images (3DEM) [12,13]. It unifies a variety of structural biology tools within a single platform, providing automation of complex workflows, interoperability between software packages and complete traceability of each protocol executed.

Scipion is an open-source tool primarily developed for Linux environments and implemented in Python programming language, ensuring flexibility, extensibility and compatibility with a wide range of scientific computing libraries. Basic installation steps are provided in APPENDIX A, while a more comprehensive guide is available at https://scipion-em.github.io/docs/release-3.0.0/docs/scipion-modes/how-to-install.html.

### *2.1.1 Modular Architecture*

Scipion is built on a plugin-based framework that enables modular integration of external computational tools [14]. This architecture provides an efficient, adaptable and customizable environment, allowing researchers to extend the platform without altering its core. Each plugin functions as an independent software module that organizes related tools and methodologies, contributing to a coherent structure for managing computational workflows.

Plugins may include various components that collectively enhance both the functionality and usability of the platform. Among these, protocols, viewers and wizards play a central role. Protocols represent the core functional units of Scipion, defining step-by-step computational procedures to execute specific algorithms or operations within a workflow. Viewers, on the other hand, are visualization tools that generate graphical

representations of output data, including 3D structures, molecular interactions and docking results. They enable users to explore and interpret complex computational outputs in a more intuitive and interactive manner. Complementing these, wizards are task-specific graphical interfaces designed to guide users through parameter selection, data input and workflow configuration. By offering real-time feedback or previews, wizards help users understand the impact of different settings before applying them to entire datasets, ultimately reducing errors, improving usability and ensuring proper protocol setup.

The modular design of Scipion allows users to install only the components they need, optimizing performance and resource use. Plugins can be added through the built-in *Plugin Manager* [12], or manually, as described in APPENDIX B. Once installed, plugins integrate seamlessly into the platform, ensuring compatibility with existing workflows and minimizing disruptions. The *Plugin Manager* provides a user-friendly interface, as illustrated in Figure 1. On the left side of the window, it displays a list of all available plugins retrieved from Scipion's servers, with installed plugins marked by filled checkboxes. On the right, two panels offer additional functionality: the upper panel presents detailed information about the selected plugin, while the lower panel contains two tabs. The first tab, labelled *Operations*, shows pending or ongoing tasks related to plugin installation or removal. The second tab, *Output Log*, records terminal output generated during these operations.

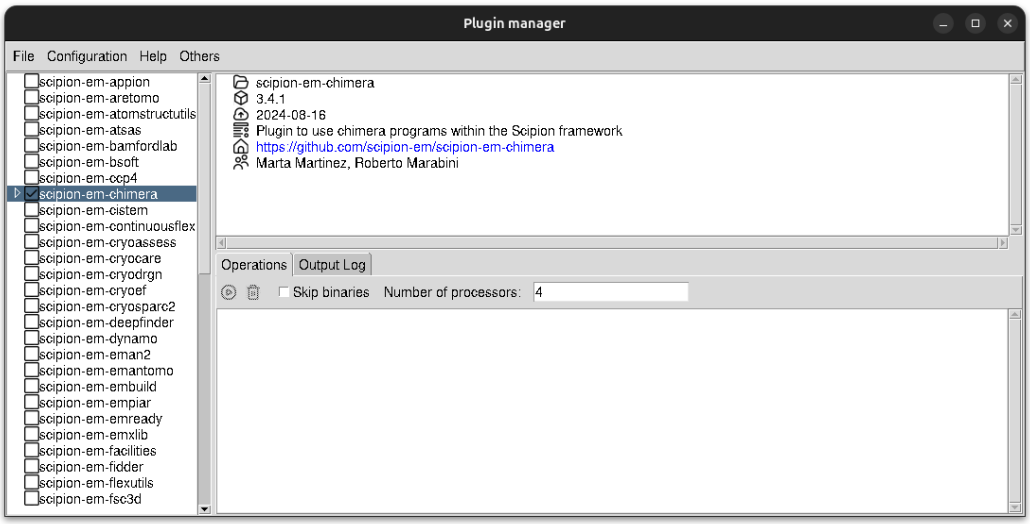



**Figure 1.** ***Plugin Manager Window.***

Installing a plugin is straightforward. Selecting a plugin's checkbox adds the installation task to the queue, while unchecking it schedules it for removal. Finally, clicking the *Run* button, represented by a circular icon with a triangle inside, executes all queued installation and uninstallation actions, streamlining the overall management of computational tools within Scipion.

### *2.1.2 Graphical User Interface (GUI)*

Scipion offers a structured and intuitive environment for managing computational workflows by organizing tasks into projects, ensuring reproducibility, automation and efficient data processing. Upon launching the software, users are presented with the *Project Manager Window*, as shown in Figure 2, which allows them to create new projects using the *Create Project* button, import existing ones via the *Import Project* button, or select from previously created projects. A *Filter* text box enables quick navigation by searching for projects by name. The interface also includes several dropdown menus for essential functionalities: the *File* menu provides access to data folders and an option to exit the framework; *Configuration* opens settings for general parameters, computing hosts and protocols; *Help* links to official documentation and displays installation information; and *Others* contains additional tools, including the *Plugin Manager*, which is essential for extending the platform's capabilities.

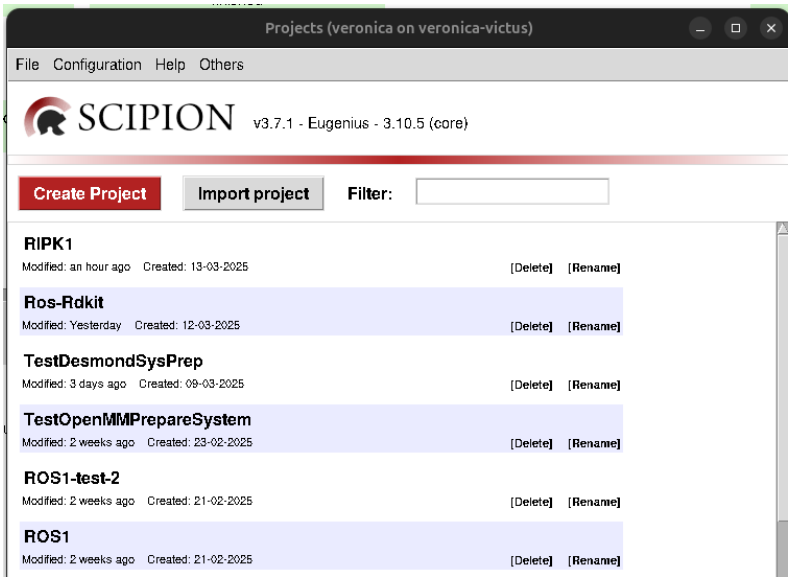



**Figure 2.** ***Project Manager Window.***

Once a project is selected, the *Project Window* becomes the main workspace for managing and executing computational workflows, as shown in Figure 3. The interface is divided into three main sections [13]. On the left, a hierarchical panel organizes installed plugins and protocols by functionality, providing quick access to tools. The central panel serves as the core workspace where workflows are built by connecting protocol boxes, which represent data flow and task dependencies. Users can switch between a list view and a flowchart view for clearer visualization. At the bottom, an information panel displays detailed data about selected protocols. This includes a *Summary* tab with input/output types and execution details; a *Methods* tab showing the protocol name and related scientific publications; a *Project Log* tab for general execution history and events; and an *Output Log* tab, further divided into *run.stdout*, *run.stderr* and *schedule.log*, which together record execution traces, error messages and task scheduling respectively, ensuring full workflow traceability and robust troubleshooting support. In addition, it includes an *Analyze Results* button, which provides access to the viewer interface associated with the corresponding protocol. This interactive feature allows users to seamlessly explore prediction outputs, enabling efficient post-processing and visualization directly within the Scipion environment.

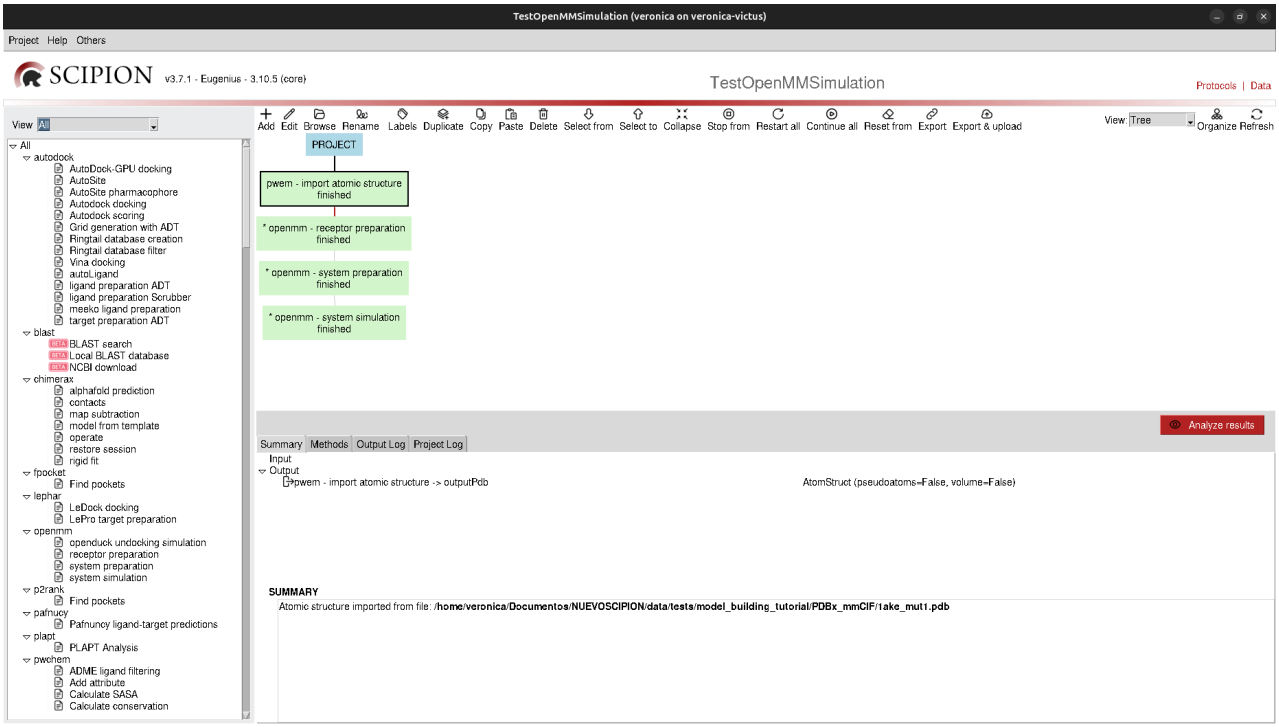



**Figure 3.** ***Project Window.***

When a protocol is added to the workflow, the *Protocol Form Window* opens (Figure 4), allowing users to configure execution parameters. While its general structure remains consistent across all protocols, the upper section includes standard fields, such as, the protocol name, optional comments, execution settings, queue management and dependencies. The lower section is protocol-specific, displaying custom fields defined by the developer, each with labels, data types, default values and additional settings. Help icons provide brief descriptions for each field. In addition, input-selection tools, such as a magnifying glass icon or a wizard icon represented by a magic wand, allow users to browse the file system or use guided interfaces for selecting files, directories or Scipion data types. For fields that support visualization, an eye icon enables real-time inspection of intermediate results, including 2D previews and 3D model visualizations.

At the bottom, three action buttons—*Close*, *Save* and *Execute*—help manage configurations, with execution starting automatically once all required inputs and dependencies are satisfied. To maintain a clean interface while supporting advanced use, the *Expert Level: Advanced* option reveals additional parameters for greater control over protocol execution.

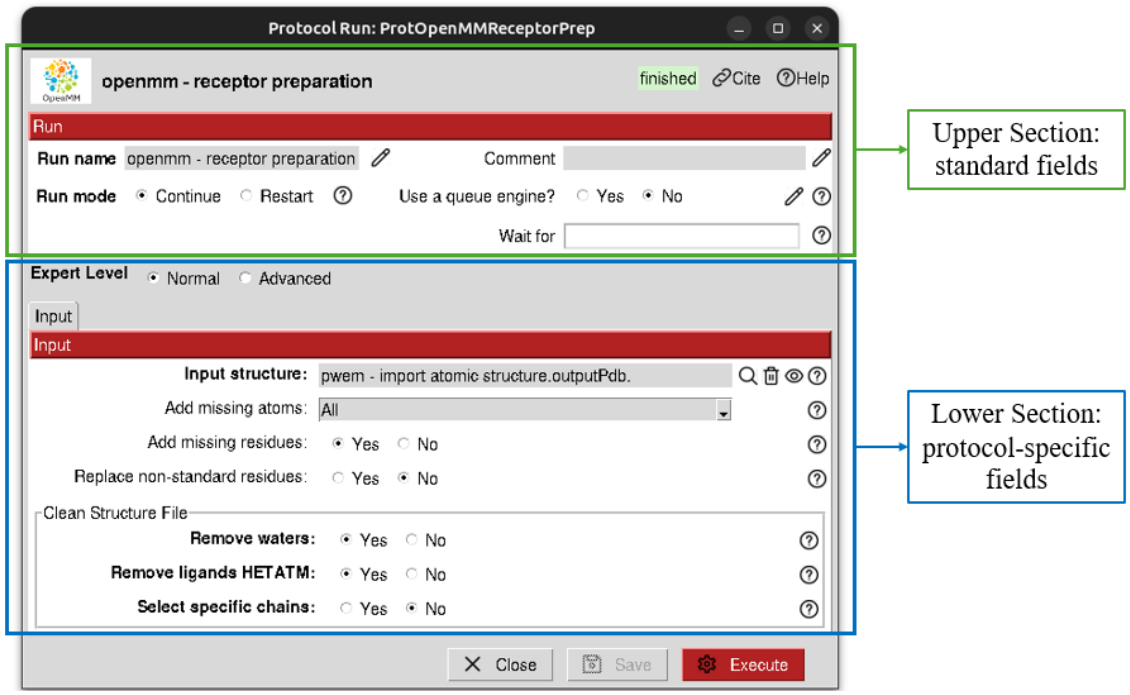



**Figure 4. Example of a *Protocol Form Window*.**

## 2.2 Scipion-chem

Scipion-chem is an advanced plugin of the Scipion workflow engine designed to optimize VDS by automating software management and enhancing interoperability across tools [15,16]. It enables seamless integration of molecular modelling software, automates file conversion and supports complete workflows encompassing ligand and receptor preparation, docking, scoring, consensus analysis and result visualization. As previously mentioned, the manual installation steps for Scipion-chem are outlined in APPENDIX B; however, the plugin can also be conveniently installed through the *Plugin Manager*.

Scipion-chem comprises around 40 protocols [16], organized into four main categories that collectively support all stages of the computational drug discovery process: general protocols, database protocols, sequence protocols and VDS protocols. General protocols handle file and object management, enabling structure conversion, metadata refinement and CSV export for external analysis.

Database protocols allow seamless interaction with public repositories, such as, UniProt and ZINC. These include the import of molecule identifiers, conversion of SMILES (Simplified Molecular Input Line Entry System) strings [17] into structured 3D formats, retrieval of UniProt cross-references [18], filtering of chemical subsets from the ZINC database [19] and the extraction of ligands associated with specific protein entries. These tools streamline data acquisition and compound selection for virtual screening.

Sequence protocols support the import and analysis of protein or nucleotide sequences from FASTA files or online databases. They include functionalities, such as, pairwise and multiple sequence alignment (MSA) [20], handling of genetic variants and definition of sequence regions of interest (ROIs), which can be mapped onto atomic structures for structural and functional evaluation [21]. These tools are particularly valuable for understanding protein conservation and mutation impact in drug discovery.

VDS protocols cover essential steps in virtual screening, including ligand and receptor preparation, molecular docking, compound filtering, scoring and consensus analysis. Ligands are prepared using RDKit or OpenBabel [16], which enable hydrogen addition, charge assignment and 3D conformer generation. However, receptor

preparation, performed with BioPython [22], ensures protein structures are cleaned, irrelevant elements are removed and missing atoms are added. To further refine candidate selection, several filtering protocols are applied: ADME (Absorption, Distribution, Metabolism and Excretion) [23] evaluates pharmacokinetic properties; PAINS (Pan-Assay Interference Compounds) [24] removes compounds prone to false positives; shape-based filtering selects molecules based on 3D geometry; and fingerprint filtering encodes molecular structures for similarity-based searches. Following this, docking and consensus protocols are used to rank potential inhibitors through pharmacophore generation, consensus analysis and rescoring. Additionally, RMSD (Root Mean Square Deviation) assesses the consistency of predicted ligand poses, consensus docking integrates outputs from multiple simulations to identify robust binders and SASA (Solvent-Accessible Surface Area) [25], calculated with BioPython, determines the extent of solvent exposure of ligands within the binding site, providing insight into binding stability.

Together, these protocols offer a robust, modular and reproducible computational pipeline for the identification, evaluation and prioritization of potential drug candidates in virtual screening projects.

## *2.3 Workflow Methodology for Virtual Drug Screening*

Having introduced the materials used, this section details the VDS workflow designed to evaluate potential ligands. To validate the workflow, we selected two well-characterized protein targets: RIPK1 (Receptor-Interacting Protein Kinase 1) and ROS1 (Proto-Oncogene Tyrosine Kinase 1).

RIPK1 is a serine/threonine kinase involved in TNF-mediated apoptosis, necroptosis and immune responses [26]. It functions through both kinase activity and scaffold roles, regulating cell death and survival. Under stress, RIPK1 activates necroptosis via RIPK3 and MLKL phosphorylation [27] and promotes pro-inflammatory cytokine production, such as, IL-6 [27].

On the other hand, ROS1 is a receptor tyrosine kinase (RTK) related to ALK and involved in oncogenic signalling through pathways such as, STAT3, PI3K/AKT and RAS/RAF/MAPK [28]. Although its physiological role is not fully understood, ROS1

gene rearrangements are common in non-small cell lung cancer (NSCLC), leading to constitutive kinase activation and tumour progression. Fusion genes like CD74-ROS1, SLC34A2-ROS1 and EZR-ROS1 drive ligand-independent signalling and uncontrolled cell growth [29].

RIPK1 and ROS1 were selected to benchmark the workflow's predictive accuracy due to their key roles in disease mechanisms and the availability of well-characterized inhibitors. These proteins provide a reliable reference framework for comparing computational predictions with experimental data. Additionally, by including both high- and low-affinity inhibitors, the workflow's ability to distinguish between strong and weak ligands can be assessed, as well as its potential to identify novel drug candidates.

For ROS1 and RIPK1, we selected 10 and 9 well-characterized inhibitors respectively from PubChem [30, 31], all of which have experimentally determined affinity values, specifically the inhibitory constant ($K_i$) and the dissociation constant ($K_d$). $K_i$ represents the concentration of an inhibitor required to reduce enzyme activity by half, while $K_d$ measures the strength of the interaction between the ligand and its target, with lower values indicating a higher binding affinity [32]. A strong binding affinity (low $K_i$ and $K_d$ values) suggests that a compound has the potential to be an effective drug, as it can tightly interact with its target at lower concentrations, thereby increasing its therapeutic efficiency.

In addition, we included a diverse set of 1300 FDA-approved drugs from the ZINC database, which were imported using Scipion without requiring any prior knowledge of their identities. Unlike the known inhibitors, these compounds lack experimental affinity data and are not necessarily strong inhibitors of the target proteins. Their inclusion allows us to assess whether the workflow can effectively differentiate the confirmed inhibitors from the broader set of FDA-approved drugs.

In VDS, particularly in molecular docking studies, the free binding energy ($\Delta G$) is commonly used as a key parameter to evaluate and rank potential drug candidates [33]. The binding energy represents the free energy change that occurs when a ligand binds to a protein, with more negative values indicating stronger interactions and a higher

likelihood of effective inhibition. It is calculated in kilocalories per mole (kcal/mol) and provides an estimate of the stability of the ligand-protein complex, guiding drug discovery efforts [33]. However, for the 19 known inhibitors from PubChem, experimental binding energy values are not directly available. Since multiple $K_i$ and $K_d$ values are reported for each ligand under different experimental conditions, we first calculated their logarithmic mean to obtain a representative affinity measure before converting it into experimental binding energy. Using the logarithmic mean instead of the arithmetic mean for affinity values is essential due to the exponential nature of binding equilibria. $K_i$ and $K_d$ values span several orders of magnitude, meaning that their distribution is typically log-normal rather than linear. Applying a logarithmic mean ensures that extremely high or low values do not disproportionately influence the average, providing a more balanced and representative measure of ligand affinity. Mathematically, the logarithmic mean of multiple $K_i$ or $K_d$ values ($K_1$, $K_2$,……, $K_n$) is given by Eq. 1:

$$LogMean = e^{\frac{1}{n}\sum_{j=1}^{n} ln\, k_j} \quad (1)$$

Once the logarithmic mean of the affinity values was obtained, we converted these values into binding free energy using the standard thermodynamic equation [33] in Eq. 2:

$$\Delta G = RTln(K_d) \quad (2)$$

where:

- $\Delta G$ is the binding free energy (kcal/mol).
- R is the universal gas constant ($1.987 \times 10^{-3}$ kcal·mol$^{-1}$·K$^{-1}$).
- T is the temperature, assumed to be physiological (298K).
- $K_d$ is the dissociation constant (measured in molar concentration, M). In this case $K_d$ is replaced by the logarithmic mean of the experimental values.

For this study, we assume that $K_i$ and $K_d$ are equivalent, as we focus solely on ligand binding without considering its effect on the substrate. This assumption is valid in cases where both constants describe binding at the same allosteric or orthosteric site, without taking enzymatic activity into account [33]. By making this simplification, we

ensure a standardized approach to binding affinity calculation while maintaining consistency across different ligands.

By calculating the experimental ΔG values beforehand, we prepared a solid reference for comparing computational predictions. In the following tables, we present the selected ligands for each target protein, including their common names. For RIPK1, due to the length and complexity of the compound names, we also include their molecular formulas. Additionally, we report the PubChem Compound ID (CID), the type of experimental affinity values available for each ligand, the logarithmic mean of these values and the resulting ΔG calculated using Eq. 2. To enhance visual interpretation, ΔG are color-coded: darker green shades represent ligands with more negative ΔG values, indicating stronger binding affinity, while red shades denote less favourable (more positive) binding energies. However, a red-coded ligand is not necessarily a poor inhibitor—it simply exhibits weaker binding relative to others in the dataset.

| **Drug Name** | **Molecular Formula** | **PubChem CID** | **Activity Type** | **Activity Value [µM]** | **Logarithmic Mean Activity Value [µM]** | **$\Delta G_{exp}$ [kcal/mol]** |
|---|---|---|---|---|---|---|
| **Gsk-3145095** | $C_{20}H_{17}F_2N_5O_2$ | 118557502 | $K_d$ | 0,00005<br>0,001 | 0,0002236 | –13,16 |
| **Methyl N-[(3S)-3-[(5-benzyl-1,2-oxazole-3-carbonyl) amino]-5-methyl-4-oxo-2,3-dihydro-1,5-benzoxazepin-7-yl] carbamate** | $C_{23}H_{22}N_4O_6$ | 118564402 | $K_i$ | 0,00028 | 0,00028 | –13,02 |
| **Necrostatin 2 racemate** | $C_{13}H_{12}ClN_3O_2$ | 643953 | $K_i$ | 0,000063<br>0,00094<br>0,002 | 0,00049 | –12,69 |
| **(R)-5-benzyl-N-(5-methyl-4-oxo-2,3,4,5-tetrahydrobenzo[b][1,4] thiazepin-3-yl) isoxazole-3-carboxamide** | $C_{21}H_{19}N_3O_3S$ | 90345378 | $K_d$ | 0,0005 | 0,0005 | –12,68 |
| **5-benzyl-N-[(3S)-5-methyl-4-oxo-2,3-dihydro-1,5-benzoxazepin-3-yl]-1,3-oxazole-2-carboxamide** | $C_{21}H_{19}N_3O_4$ | 137632484 | $K_i$ | 0,0004<br>0,0005<br>0,0016 | 0,0006839 | –12,50 |

| | | | | | | |
|---|---|---|---|---|---|---|
| **(3S)-3-(2-benzyl-3-chloro-7-oxo-4,5-dihydropyrazolo[3,4-c] pyridin-6-yl)-5-methyl-4-oxo-2,3-dihydro-1,5-benzoxazepine-7-carbonitrile** | $C_{24}H_{20}ClN_5O_3$ | 135309066 | $K_d$ | 0,0008511 | 0,0008511 | –12,37 |
| **RIPK1-IN-15** | $C_{19}H_{19}N_3O_2$ | 162385864 | $K_d$ | 0,00032<br>0,01 | 0,001788 | –11,93 |
| **RIPK1-IN-10** | $C_{30}H_{28}F_2N_6O_4$ | 164946835 | $K_i$ | 0,0005<br>0,004<br>0,01 | 0,00271 | -11,68 |
| **1-[3-(3-Fluorophenyl)-3,4-dihydropyrazol-2-yl]-2,2-dimethylpropan-1-one** | $C_{14}H_{17}FN_2O$ | 122703743 | $K_d$ | 3,97<br>3,55 | 3,75413 | -7,39 |

**Table 1. Selected RIPK1 ligands with experimental affinity data and calculated binding energies.**

| Drug Name | PubChem CID | Activity Type | Activity Value [µM] | Logarithmic Mean Activity Value [µM] | $\Delta G_{exp}$ [kcal/mol] |
|---|---|---|---|---|---|
| **Entrectinib** | 25141092 | $K_i$ | 0,00002<br>0,000025<br>0,0001 | 0,000037 | -14,23 |
| **Brigatinib** | 68165256 | $K_i$ | 0,00007<br>0,000145<br>0,0002<br>0,001 | 0,000212 | -13,19 |
| **Ceritinib** | 57379345 | $K_i$ | 0,00017<br>0,00007<br>0,0084 | 0,000464 | -12,73 |
| **Staurosporin** | 44259 | $K_i$ | 0,0019 | 0,0019 | -11,90 |
| **Crizotinib** | 11626560 | $K_i$ | 0,007<br>0,005<br>0,0002<br>0,012 | 0,003027 | -11,62 |
| **Nintedanib** | 135423438 | $K_i$ | 0,0004<br>0,0007<br>0,23 | 0,004 | -11,45 |
| **Foretinib** | 42642645 | $K_i$ | 0,0006<br>0,0058<br>0,33 | 0,01047 | -10,89 |

| | | | | | |
|---|---|---|---|---|---|
| **Dasatinib** | 3062316 | $K_i$ | 0,0082<br>0,016<br>0,02 | 0,013793 | -10,72 |
| **Lorlatinib** | 71731823 | $K_i$ | 0,0082<br>0,02<br>0,018 | 0,01434 | -10,69 |
| **Repotrectinib** | 135565923 | $K_d$ | 3,7 | 3,7 | -7,41 |

**Table 2. Selected ROS1 ligands with experimental affinity data and calculated binding energies.**

### *2.3.1 Step-by-Step Execution of the VDS Workflow*

The VDS workflow developed for this study consists of a sequence of well-defined steps aimed at preparing, analysing and evaluating both target proteins and small molecule ligands. Each step corresponds to a specific protocol within the Scipion-chem environment and the overall execution order has been carefully structured to ensure methodological reproducibility and consistency across the dataset. In the following subsections, each workflow step is described in detail, including its objective and the specific parameters used for execution.

Before addressing each step in detail, a visual overview of the complete VDS workflow is presented in APPENDIX C due to space constraints. Figure Appendix 1 illustrates the main stages and their interconnections, providing a comprehensive perspective on the data flow from the initial inputs—namely, target proteins and ligand libraries—to the final outputs. These outputs include docking scores and molecular dynamics-derived metrics, which can support informed decision-making in the context of drug discovery.

While some steps, such as data import and structural preparation, are considered fundamental and common to most computational workflows, the core contribution of this methodology lies in how molecular filtering strategies are integrated with consensus protocols and ligand ranking processes.

#### *2.3.1.1 Molecule Import*

The first step of the workflow involves importing both small molecule ligands and the target protein structures into the Scipion environment. This preliminary setup ensures that all structural data are available for subsequent analysis. Two specific protocols were used for this purpose: *Import Small Molecules*, provided by the Scipion-chem plugin, and *Import Atomic Structure*, available through the Scipion-em plugin.

The *Import Small Molecules* protocol allows integration of compound libraries either from user-provided files or directly from predefined databases such as ZINC or ECBL. In this show case, the set of 1300 FDA-approved drugs were imported directly from the ZINC database. To facilitate this, the parameter *Format molecules from smiles* was set to *Yes*, allowing the protocol to interpret the SMILES strings and convert them into 3D structures and choosing RDKit as the structure manager—preferred over OpenBabel, which is currently deprecated. As ZINC already provides 3D-optimized molecular structures, the *Optimize 3D structure* option was set to *No*, avoiding unnecessary geometry refinement. The complete configuration of this protocol can be more clearly observed in Figure 5, which illustrates the setup used during the import process of the ligands.

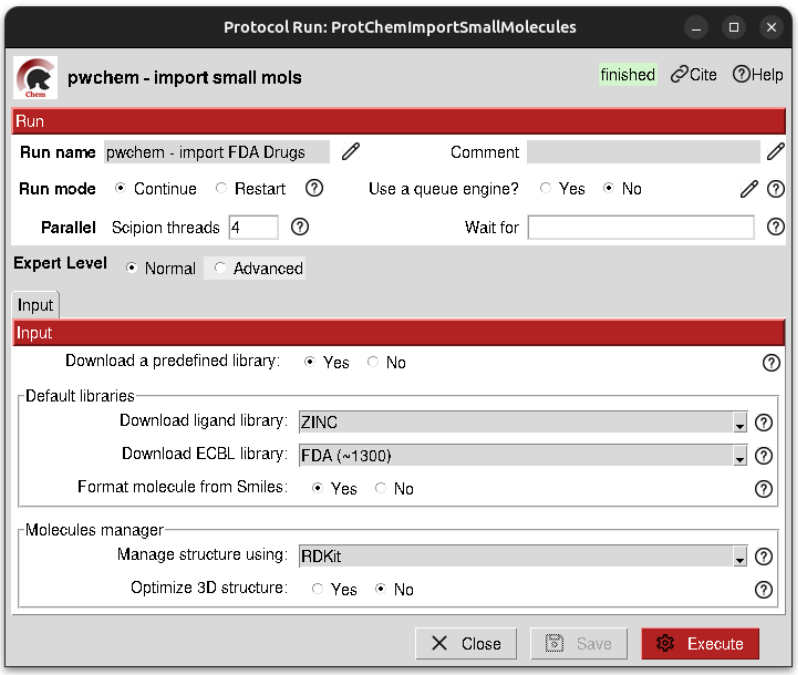



**Figure 5. *Protocol Form* configuration for importing FDA-drugs from the ZINC database.**

To complement the FDA drug set, an additional group of well-characterized inhibitors per target protein was included. These molecules were manually downloaded from PubChem [30, 31] in SDF format, although the protocol also supports formats such as *.smi*, *.mol2*, *.mae*, and *.pdb*. These files were imported using the same *Import Small Molecules* protocol. The corresponding *Protocol Form* was configured to specify the local directory path and the file naming pattern for batch import. Again, RDKit was selected for structure processing and geometry optimization was turned off, since the downloaded SDF files already contained valid 3D conformations. The full setup is illustrated in Figure 6.

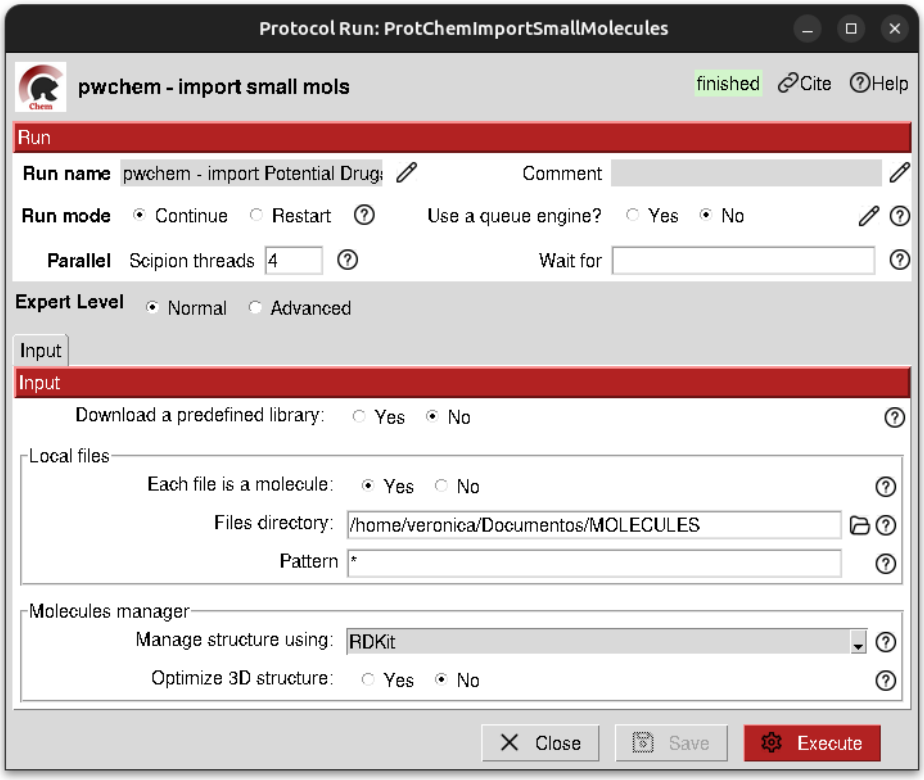



**Figure 6. *Protocol Form* configuration for importing potential known ligands from local SDF files.**

The *Import Atomic Structure* protocol was employed to incorporate the target protein structures into the workflow (Figure 7). For ROS1, the structure was retrieved automatically from the Protein Data Bank (PDB) using its PDB ID 3ZBF [31], thereby streamlining the import through direct database integration. In contrast, the structure of RIPK1 was previously downloaded in PDB format from PubChem [30] and imported as a local file by specifying the appropriate path during protocol configuration.

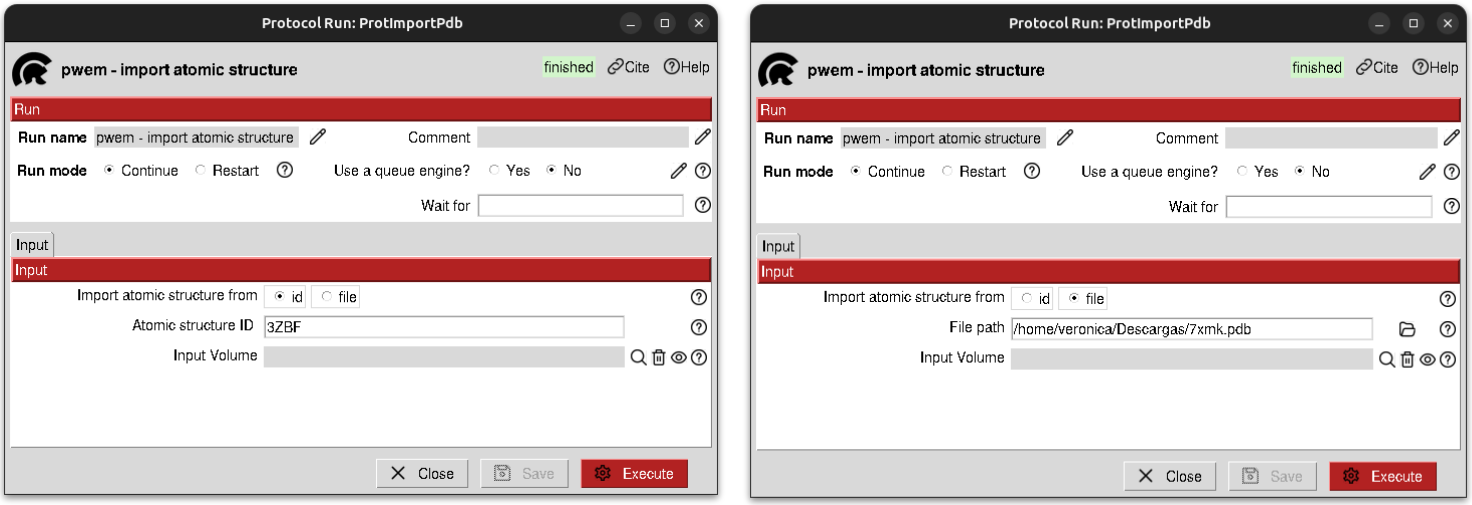



**Figure 7. *Protocol Form* configuration for importing ROS1 (left) and RIPK1 (right).**

### *2.3.1.2 Molecule Preparation*

Before initiating the structure preparation protocols, all ligands intended for each target protein were unified into a single set. This step was necessary because the molecules, FDA-approved drugs from ZINC and known inhibitors from PubChem, had been initially imported through separate protocol configurations, resulting in two distinct sets within the workflow. Although optional, this unification step simplifies subsequent protocol application and ensures that ligand preparation and analysis are carried out consistently across all compounds.

To merge the two sets, the *Operate Set* protocol from the Scipion-chem plugin was employed. This protocol offers a range of functionalities to manipulate objects via their internal SQLite representation. Among the available operations—such as *Unique*, *Union*, *Intersection*, *Difference*, *Filter*, *Remove Column* and *Ranking*—the *Union* operation was selected to combine the ligand sets corresponding to each target protein. This process integrates all elements from both sets into a single, unified collection, ensuring that all compounds are included and prepared for subsequent stages of analysis. In this case, duplicate removal was not necessary, as no overlap existed between the imported datasets. The reference attribute used to define the merging criterion was the ligand name: for FDA drugs, this corresponded to their ZINC ID, while for the known inhibitors from PubChem, it matched the common name as presented in Table 1 and Table

2. The resulting unified set constitutes a comprehensive compound library per protein, combining both benchmark inhibitors and a broader spectrum of FDA-approved molecules for virtual screening. The full configuration of this protocol, along with the list of available attributes, is shown in Figure 8.

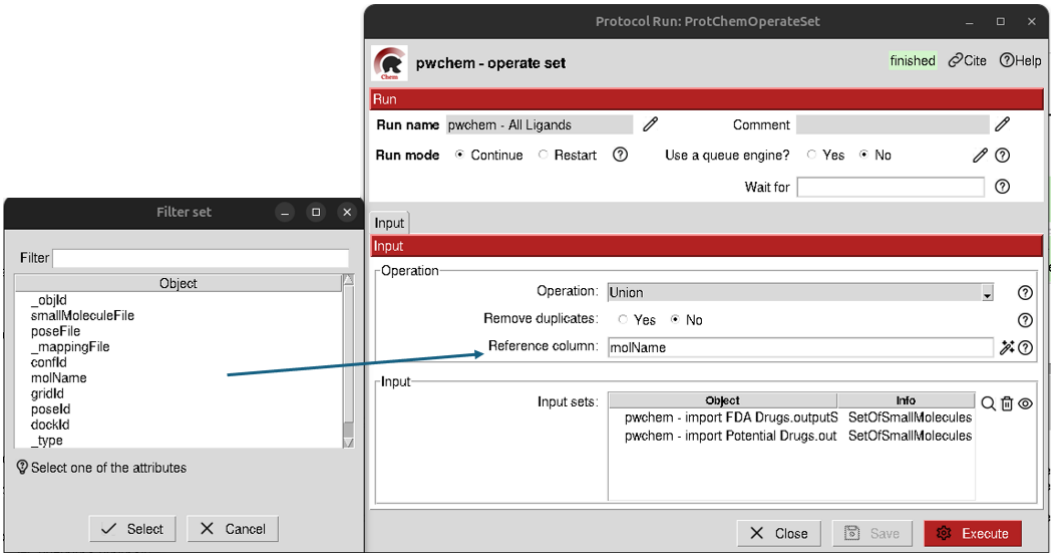



**Figure 8.** ***Protocol Form*** **configuration for merging ligand sets using the** ***Operate Set*** **protocol.**

Once the structures were successfully imported into the Scipion workflow and all ligands were consolidated into a single set, separate preparation steps were carried out for the protein targets and the ligand libraries to ensure compatibility with subsequent protocols.

Protein structures obtained from databases may contain elements that are unnecessary or even detrimental for downstream computational analyses, such as water molecules, heteroatoms or non-relevant protein chains. Additionally, they may lack hydrogen atoms or include incomplete residues, which can compromise the accuracy of docking or molecular dynamics simulations. To address these potential issues, the *Target Preparation* protocol was applied. This protocol integrates PDBFixer, a molecular modeling tool that facilitates preprocessing of protein structures by removing undesired components [34], preserving specific chains and completing missing atoms or residues when necessary. In this case, the configuration was deliberately kept minimal, as illustrated in Figure 9. PDBFixer was disabled, as no missing atoms or problematic residues had been identified in the structures of ROS1 or RIPK1. Similarly, the option to

retain a specific protein chain was also deactivated, as no chain-level selection was required. However, the options to remove water molecules and remove heteroatoms were enabled to eliminate non-essential components from the structures prior to docking. This protocol was selected over alternatives, such as *LePro Target Preparation* (Lephar plugin), *Target Preparation ADT* (AutoDock plugin) or *Target Preparation (prepwizard)* (Schrödinger plugin), due to its native availability within the Scipion-chem plugin, eliminating the need to install or configure additional plugins. Moreover, alternative protocols often involve a larger number of chemical and structural parameters, requiring a deeper understanding of protein structure refinement and force field selection.

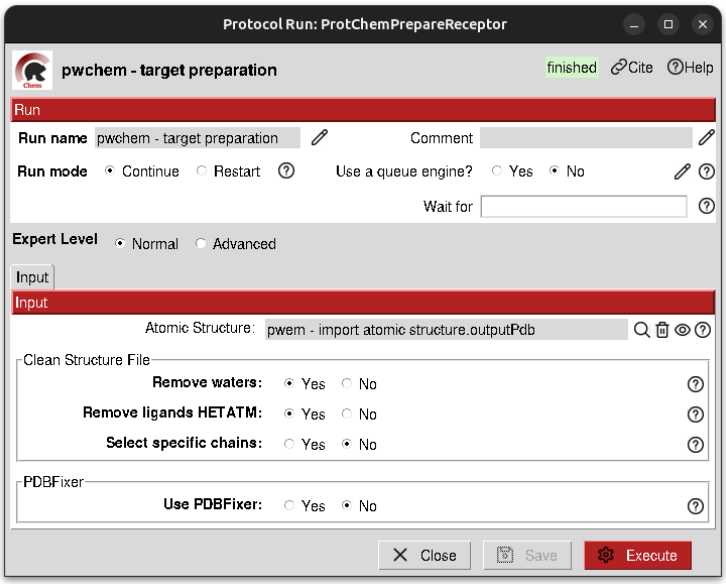



**Figure 9. *Protocol Form* configuration for the target protein preparation.**

Ligands also require thorough preparation to ensure chemical correctness and compatibility with docking protocols. Scipion offers multiple protocols for ligand preparation, including *Ligand Preparation ADT*, *Meeko Ligand Preparation* and *Ligand Preparation Scrubber* (from the AutoDock plugin), *Ligand Preparation LigPrep* (from the Schrodinger plugin), as well as protocols based on RDKit and OpenBabel within the Scipion-chem plugin. Among these, *RDKit Ligand Preparation* (Figure 10) was selected for this study due to its broad applicability, efficiency, independence from external docking suites and consistency with the previous import step, where RDKit was also used for structure processing. The RDKit preparation protocol was configured to reassign hydrogen atoms, ensuring accurate protonation states that reflect the chemical and

physiological properties of each ligand. This step is essential for maintaining structural validity and for correctly modelling potential hydrogen bonding interactions during docking. Additionally, Gasteiger partial charges were recalculated, as these charges significantly influence electrostatic interactions with the target protein and are, therefore, critical for reliable docking outcomes. For this calculation, the MMFF94 force field was selected due to its broad applicability and proven accuracy for drug-like molecules. Although the protocol also offers the use of MMFF94s, a variant fine-tuned for certain functional groups like $sp^2$-hybridized nitrogens [35], MMFF94 was chosen for its general robustness across a wider range of chemical scaffolds. The option to generate conformers was disabled in this case, primarily due to the large number of molecules processed. Generating multiple conformers per molecule increases computational demand significantly. However, in scenarios involving smaller ligand libraries, enabling this feature is strongly recommended, as it enhances docking performance by exploring multiple low-energy 3D conformations, thereby increasing the likelihood of identifying optimal binding poses.

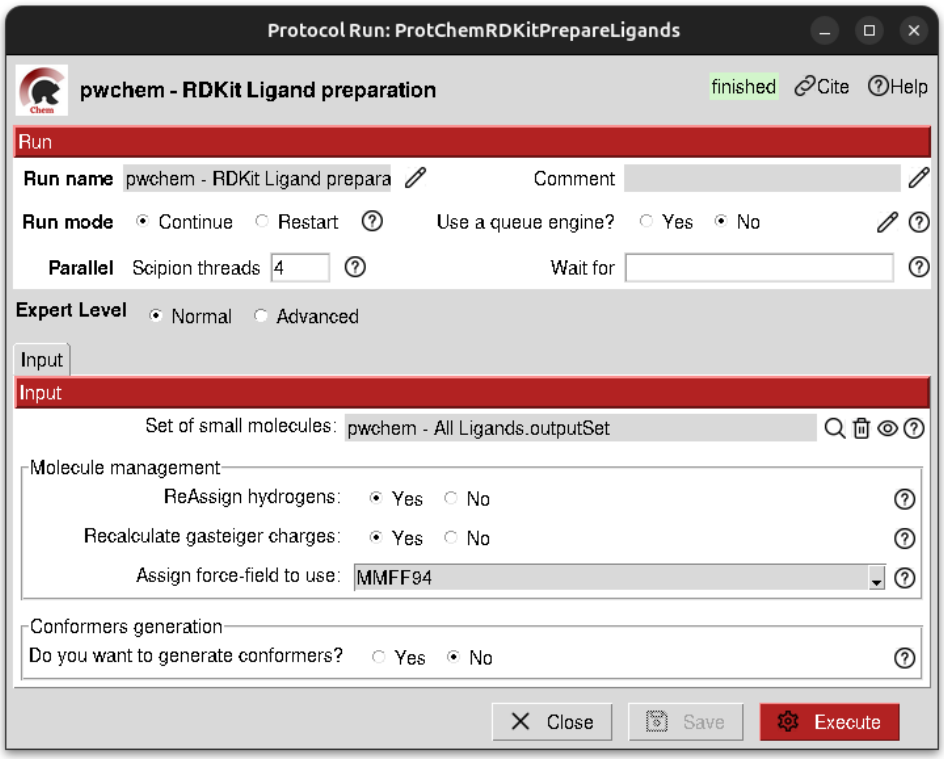



**Figure 10.** ***Protocol Form*** **configuration of the** ***RDKit Ligand Preparation*** **protocol.**

#### *2.3.1.3 ROI Definition and Structural Consensus of ROIs*

Defining the regions of interest within a target protein is a fundamental step. These ROIs typically correspond to potential binding pockets or interaction sites, allowing docking protocols to focus on the most relevant areas instead of the entire protein surface. In Scipion-chem, ROIs are defined as atom-level groups within the protein structure and can be located directly on atoms, on the molecular surface or in the surrounding space. Their main function is to reduce the conformational search space, enhancing the precision and efficiency of docking and filtering protocols. Additionally, ROIs may serve broader purposes within the platform, such as mapping known mutations or analysing functional residues.

Scipion-chem offers several protocols for ROI definition, which can be grouped into four categories. First, ROIs can be defined manually by selecting specific residues or atoms based on coordinates, existing ligands, known functional sites or interchain interfaces. Second, ROIs can be determined based on structural properties of the protein, such as selecting residues with high SASA. Third, Scipion-chem includes integration with several pocket prediction tools that identify likely ligand-binding regions. Some of these tools are Fpocket, which uses geometric cavity analysis through alpha-spheres and evaluates pocket properties; P2Rank, which applies machine learning trained on surface features to predict ligand sites; and AutoSite, which uses grid-based energy scoring to identify regions with favourable interaction energies [8]. Lastly, ROIs can also be defined from protein sequences, based on conservation, natural variants or user-defined interest. These sequence-based ROIs can be mapped onto structural models for spatial analysis.

In this study, although binding pockets for ROS1 and RIPK1 have been previously described in the literature [36,37], multiple pocket prediction tools were employed to test the available protocols and to showcase the flexibility and integrative capacity of Scipion-chem for defining ROIs. Rather than manually specifying predefined ROIs—often narrowly focused on published studies—the decision was made to predict them computationally to ensure a more exploratory and comprehensive analysis. Three pocket prediction tools were applied: Fpocket, P2Rank and AutoSite, each relying on distinct theoretical approaches as previously described. This multi-tool strategy enhances the

robustness of the analysis by increasing confidence in the identification of druggable regions.

To configure the three pocket prediction protocols, different parameterization strategies were applied depending on the target. For ROS1, the parameters were adapted from a previously published study [38] that used similar tools, ensuring alignment with established methodologies. In contrast, for RIPK1, no literature references were found employing these specific tools, so the default parameters provided by Scipion-chem were used. This approach highlights the platform's flexibility in adapting to both well-studied and less-characterized targets. Among the tools used, P2Rank offers the most accessible setup, requiring only the input of the prepared protein structure, making it especially suitable for users with limited prior knowledge of the target. In comparison, AutoSite requires manual configuration of the step size (the distance between points in the electrostatic grid) and the number of neighbours, which defines the minimum number of adjacent grid points needed for a region to be identified as a pocket. Fpocket, on the other hand, requires the specification of parameters related to alpha-sphere detection and clustering criteria, which may require deeper understanding of the underlying pocket prediction algorithm. The complete configuration details for each of these protocols are illustrated in Figure 11 and Figure 12.

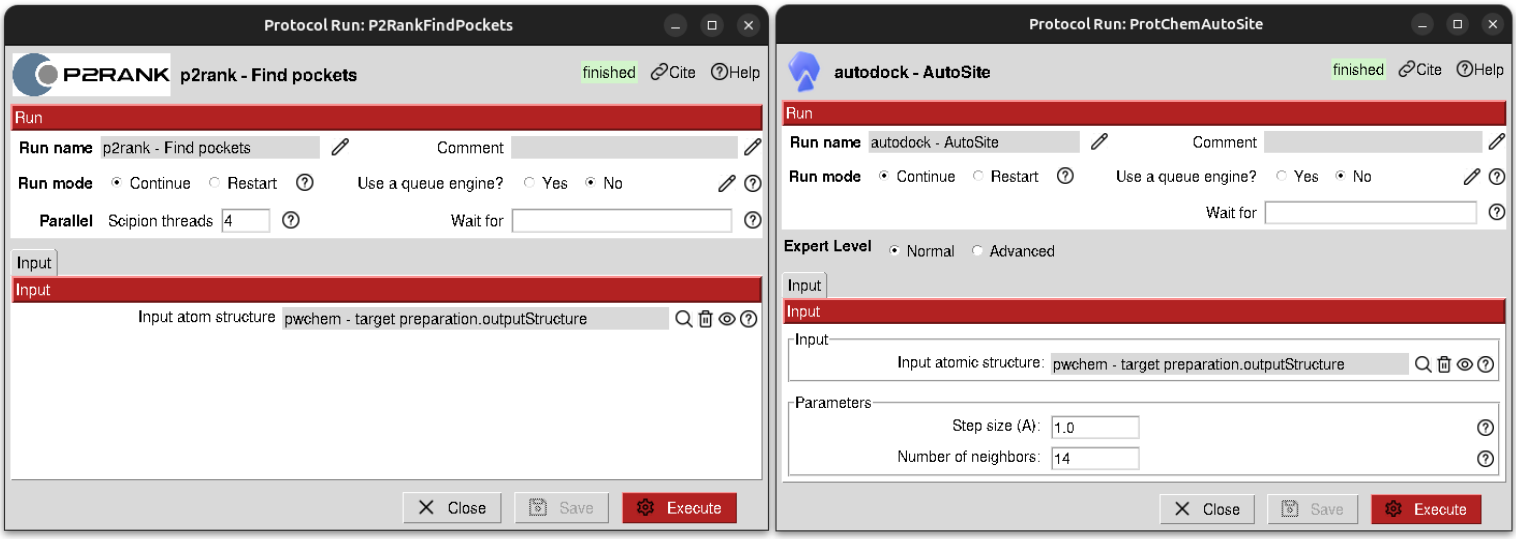



**Figure 11. Configuration of P2Rank (left) and AutoSite (right) protocol for ROS1 and RIPK1.**

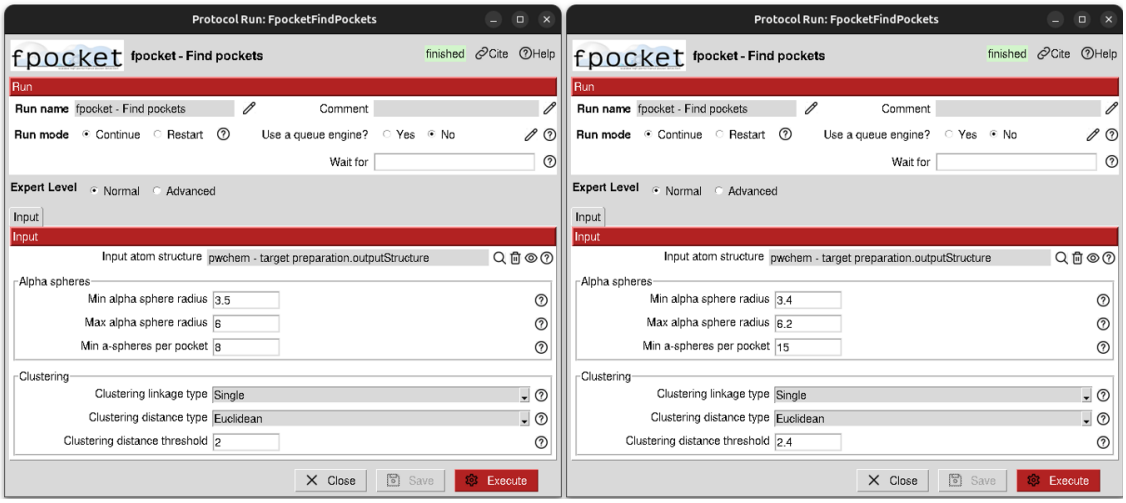



**Figure 12. Configuration of the *Fpocket Find Pockets* protocol for ROS1 (left) and RIPK1 (right).**

After obtaining the binding site predictions from the selected tools, the next essential step is to integrate these results using the *Consensus ROI* protocol available. This protocol consolidates the predicted ROIs by comparing multiple input sets and retaining only those regions that appear consistently across different methods. It clusters ROIs based on residue overlapping, identifying shared or partially overlapping binding pockets. For this study, the protocol was configured so that two ROIs are considered overlapping if they share a sufficient proportion of involved residues, using the default distance threshold provided by Scipion-chem to define spatial proximity (Figure 13). By applying this consensus analysis, the workflow filters out tool-specific artifacts and highlights the most robust and druggable regions.

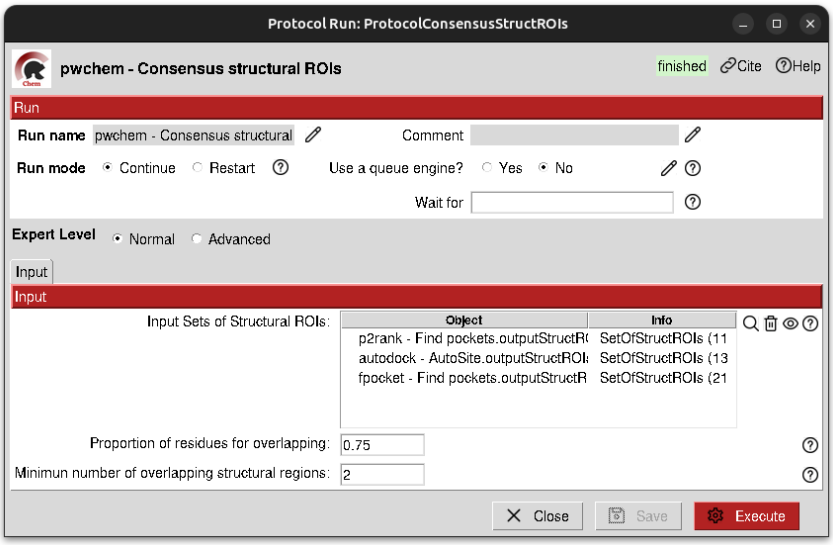



**Figure 13. Configuration of the *Consensus Structural ROIs* protocol.**

#### *2.3.1.4 Docking*

This step involves executing molecular docking simulations using three independent docking programs: AutoDock-GPU, AutoDock and LeDock. These protocols are applied to the consensus predicted ROIs and the prepared ligands using default parameters for both target proteins. Docking represents one of the most computationally demanding stages in a VDS workflow and often becomes a performance bottleneck, particularly when processing many ligands, such as, those used in this study. Therefore, appropriate allocation of computational resources is critical. The docking protocols allow users to specify the number of CPU threads and in the case of AutoDock-GPU, the number of GPUs to use, providing significant acceleration over traditional CPU-based execution (Figure 14). Although AutoDock-GPU offers clear advantages in terms of performance and scalability, the inclusion of AutoDock (CPU version) and LeDock adds valuable methodological diversity and strengthens the robustness of the workflow. AutoDock remains one of the most widely validated docking programs in computational chemistry [5]. LeDock, on the other hand, is a fast and resource-efficient docking tool, noted for its ability to rank ligands accurately and generate diverse conformational poses with minimal computational cost [39]. By employing three distinct docking engines, the workflow benefits from a multi-perspective evaluation of ligand-receptor interactions. This approach reduces the potential bias of relying on a single algorithm and enables a consensus scoring strategy based on diverse methodologies.

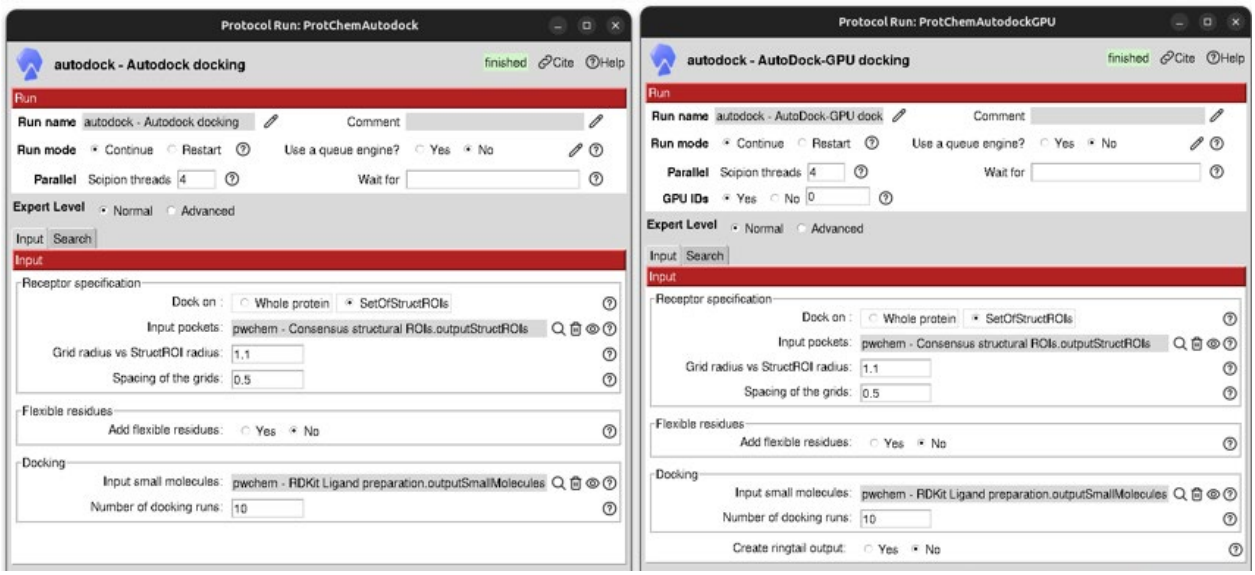



**Figure 14. Configuration of docking protocols: AutoDock-GPU (right) and AutoDock (left).**

#### *2.3.1.5 First Filtering*

The output of each docking protocol consists of a set of docked poses, with each ligand typically associated with multiple poses, each representing a potential binding configuration within the target's binding site. Every pose includes a predicted binding energy, which reflects the estimated strength of the ligand-target interaction. Although the specific energy scoring functions differ across docking engines, the general principle remains the same: the more negative the binding energy, the more favourable the predicted interaction. To streamline the dataset and prioritize the most promising candidates, a preliminary filtering step was applied independently to the output of each docking protocol. For this purpose, the *Operate Set* protocol, previously introduced during the molecule preparation step, was used in filtering mode. The filtering criterion was straightforward: retain only those poses with strictly negative binding energy values [33].

Since positive binding energies correspond to thermodynamically unfavourable interactions, these poses were excluded from further analysis. This initial filtering step effectively reduces the number of poses carried forward, ensuring that only the most plausible binding configurations are considered in subsequent stages of the VDS workflow.

#### *2.3.1.6 Score Docking Poses*

To ensure consistent comparison across docking results, all filtered poses with negative binding energies were rescored using the *ODDT Score Docking* protocol, which leverages the Open Drug Discovery Toolkit (ODDT). Among the available scoring functions—Vina, RFScore (Random Forest Score), NNNScore (Nearest Neighbour Neural Network Score) and PLECScore (Protein–Ligand Extended Connectivity Score)—the Vina score (Figure 15) was selected due to its wide compatibility, computational speed and reliance on interaction features, such as, steric fit, hydrogen bonding, hydrophobic contacts and torsional penalties.

Unlike the machine learning–based alternatives, Vina does not require target-specific training, making it particularly suitable for rescoring large pose sets in a

standardized manner. As a result of this step, the same docking poses were retained, but each now includes a unified attribute applied consistently across all docking engines.

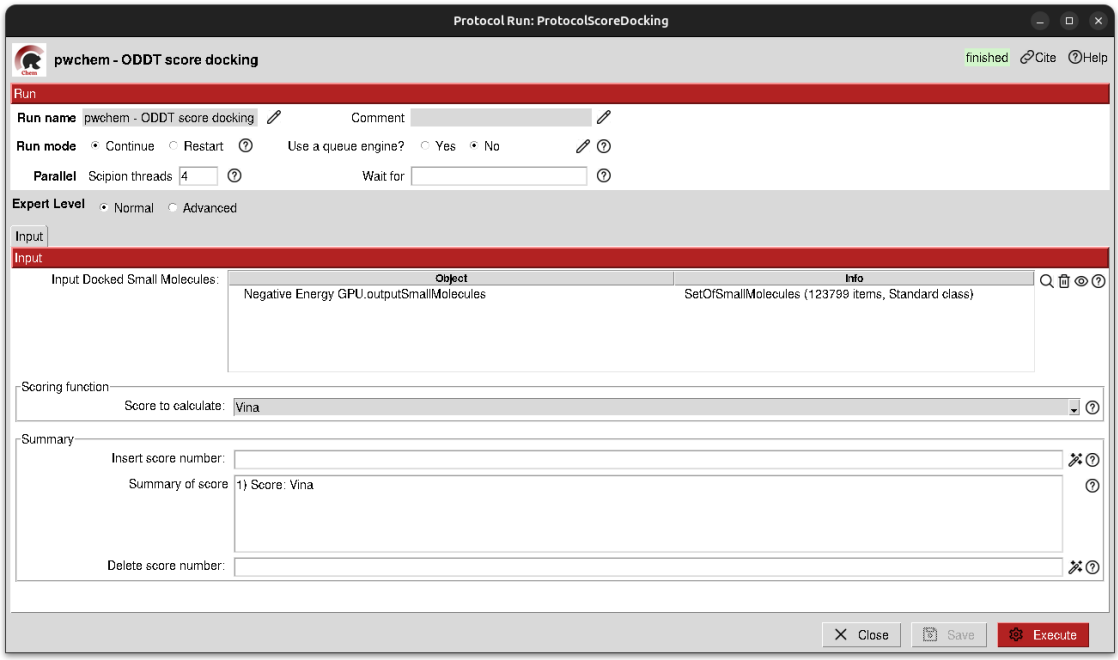



**Figure 15. Configuration of the *ODDT Score Docking Protocol* for Vina rescoring.**

### *2.3.1.7 Second Filtering*

To further narrow down the docking poses to the most promising ligand–target interactions, a second filtering step was performed, but this time based on the rescoring values generated by the ODDT Vina score. Unlike binding energies, where more negative values indicate stronger binding affinity, the Vina score interprets higher values as better predicted interactions.

Using the *Operate Set* protocol in filtering mode, only poses with positive Vina scores were retained, while those with negative scores indicating unreliable interactions, were discarded. This approach was adopted following recommendations from the Scipion-chem development team, who advise filtering out negative-scoring poses to avoid artefactual ligand–receptor associations. This filtering was applied independently to the outputs of AutoDock-GPU, AutoDock and LeDock, refining each dataset to focus on high-confidence docking results.

### *2.3.1.8 Consensus Docking*

To integrate the docking results from AutoDock-GPU, AutoDock and LeDock, a *Consensus Docking* protocol was applied over the filtered sets of poses. This protocol clusters individual ligand conformations based on their three-dimensional atomic similarity, using RMSD between atomic coordinates as the primary metric. Two clustering strategies are available: the default method uses the Scipy library to compute all pairwise RMSD distances between poses, ensuring high clustering accuracy at the cost of greater computational effort; the alternative method accelerates the process by comparing each new pose only against the representative pose of each existing cluster reducing the number of RMSD calculations. However, this method is order-dependent, as the outcome may vary depending on the sequence in which poses are processed. After clustering, a single pose is selected from each cluster as the representative, typically the one with the lowest binding energy or highest score, depending on the scoring scheme.

To assess the robustness of the consensus approach, two separate consensus configurations were performed (Figure 16): one uses their predicted docking energies and the other their rescored Vina values. In both cases, only clusters containing poses from at least two out of the three docking programs were retained. Additionally, the maximum RMSD threshold for cluster overlaps and the hierarchical clustering linkage method were kept at their default values, as these parameters are well-supported in the literature for producing reliable and consistent clustering results [40].

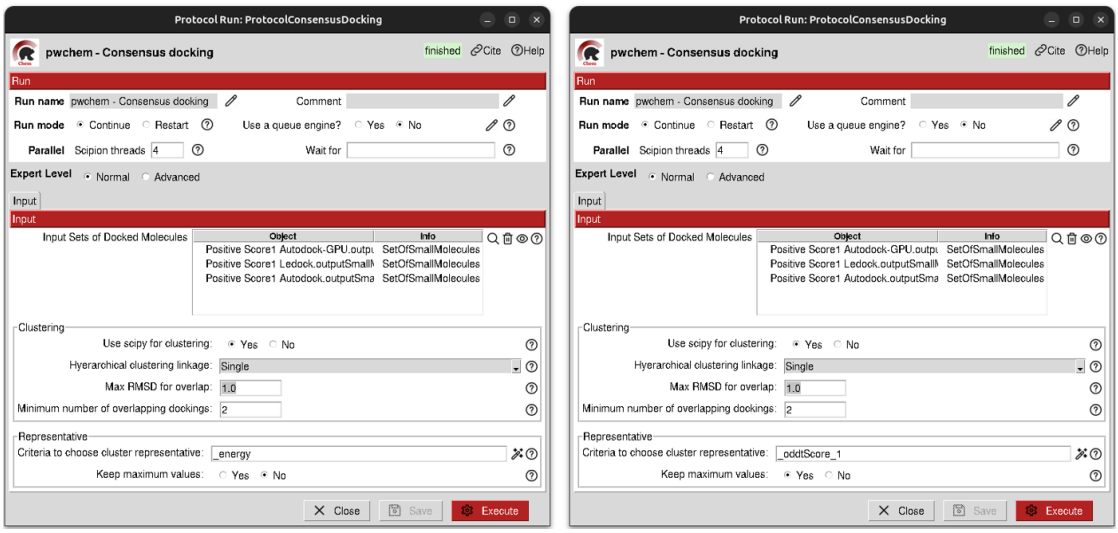



**Figure 16. Configuration of *Consensus Docking* protocols based on binding energy (left) and Vina score (right).**

The dual consensus executions assess the protocol's performance using two scoring schemes: the Vina score, uniformly calculated across all docking protocols, and the predicted binding energy, which varies with each docking engine. This comparison helps determine which approach better selects representative poses and preserves therapeutically relevant ligands.

#### *2.3.1.9 Third Filtering*

After completing the consensus docking step, a final filtering stage was performed using the *Operate Set* protocol to further refine the pool of docking poses. In this third filtering step, only poses with a predicted binding energy lower than −7 kcal/mol were retained. Although poses with slightly less favourable energies may still represent plausible interactions, the objective at this stage was to isolate only the most promising candidates with the strongest predicted affinities.

The threshold of −7 kcal/mol was chosen because ligands with binding energies below this value are generally considered to exhibit high binding affinity and are, therefore, more likely to perform well in downstream experimental validation [33]. In addition, this cutoff aligns with the binding energy range observed in the weakest of the known inhibitors initially included in the study.

#### *2.3.1.10 Ranking of Docking Scores*

After the third filtering step, the *Rank Docking Score* protocol is applied, which is designed to support the prioritization of candidate molecules by evaluating all available docking poses for each ligand. This protocol first groups all poses by their common ligand name and then selects the optimal pose within each group based on a user-defined criterion.

As with the consensus step, the protocol will be evaluated under different configurations (Figure 17), ranking the ligands based either on predicted docking energy or on Vina score. In addition, the protocol assigns a *rankScore* to each ligand, reflecting its priority based on its performance across multiple docking rankings. A higher *rankScore* indicates that the ligand consistently ranked well across different scoring lists,

providing a unified and interpretable candidate list for downstream analyses like MD simulations.

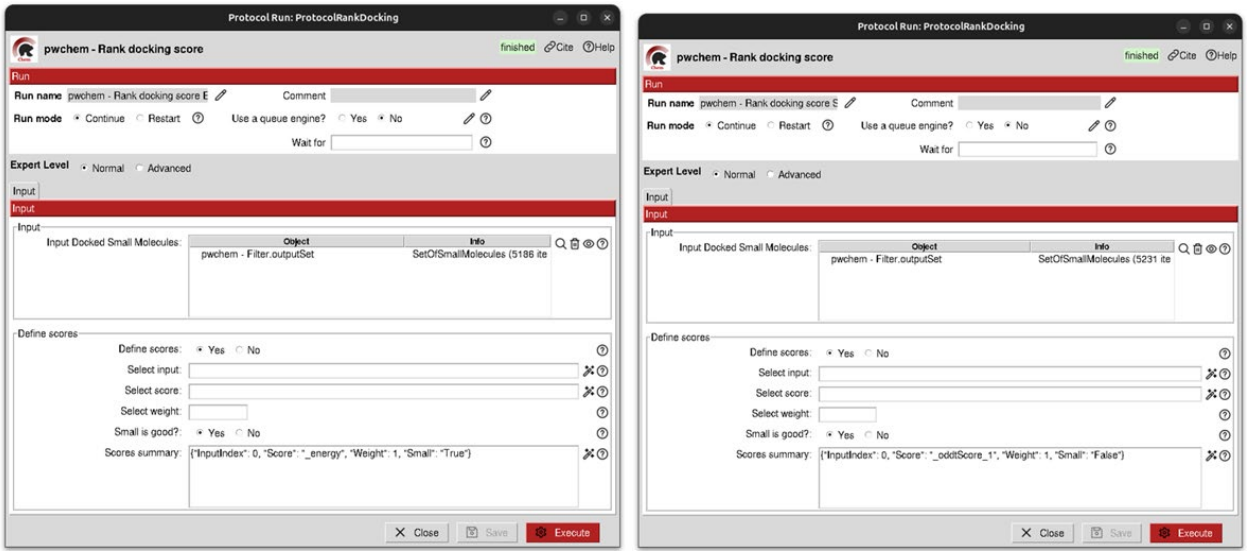



**Figure 17. Configuration of *Ranking Docking Score* protocols based on binding energy (left) and Vina score (right).**

### *2.3.1.11 Molecular Dynamics*

The final step of the VDS workflow involves molecular dynamics simulations to validate the stability and plausibility of the predicted ligand–target interactions. MD is a powerful computational technique that simulates the atomic-level motion and interactions of molecules over time, offering a dynamic and more realistic representation of binding behaviour compared to static docking methods. However, due to its high computational cost, applying MD to all docked ligands is generally unfeasible.

Instead, MD is best reserved for a small subset of compounds that have already passed stringent filtering criteria. In this study, simulations were selectively performed on the top-ranked and lowest-ranked reference ligands (according to the experimental energy in Table 1 and Table 2) for each target protein. This approach enables direct assessment of whether the most promising candidate maintains stable interactions within the binding site and whether the weakest candidate fails to do so—thus validating the docking-based ranking.

Scipion-chem integrates several MD engines for system preparation and simulation, including GROMACS, AmberTools, OpenMM and Schrödinger. In this case, Schrödinger was chosen over other MD engines because, unlike some alternatives integrated in Scipion-chem, it allows the user to extract results specifically for the ligand, not just the target protein.

Molecular dynamics in Scipion-chem requires the sequential execution of two dedicated protocols (Figure 18): one for system preparation and another for running the simulation. First, the *Schrödinger System Preparation (Desmond)* protocol constructs the solvated system, defining the solute boundary box, selecting the force field and adding ions either to neutralize the system or to match a desired ionic concentration. In this study, all preparation parameters were maintained at their default values to ensure standardization. Following this, the *Schrödinger MD Simulation (Desmond)* protocol was used to perform molecular dynamics on the prepared systems. While the protocol interface supports the definition of multiple sequential stages, including energy minimization, equilibration and production runs, this study employed the default simulation scheme, *NPT Desmond Relaxation*, which executes a standard relaxation process. This default configuration was chosen for its reliability and consistency across systems.

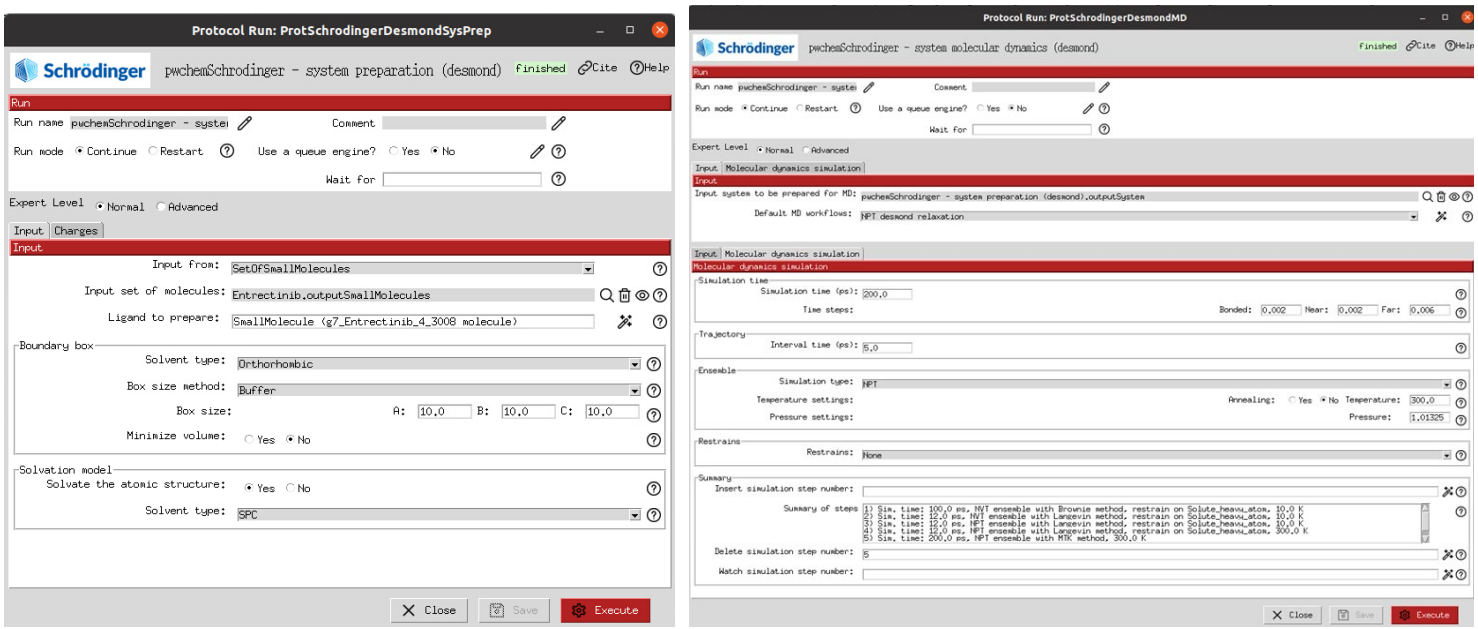



**Figure 18. Configuration of *Schrödinger MD Simulation (Desmond)* protocol (left) and of *Schrödinger System Preparation (Desmond)* protocol (right).**

By comparing the dynamic behaviour of the best and worst ligands, particularly their ability to remain stably bound in the receptor pocket, this step offers an additional level of validation. It enhances the biological relevance of the VDS results and helps prioritize candidates with not only favourable docking scores, but also robust interaction profiles under dynamic, physiologically relevant conditions.

Throughout the development and execution of the VDS workflow, several issues were encountered that impacted the pipeline. Although the major steps of the workflow were successfully implemented, multiple failures and unexpected behaviours emerged during their practical execution. All identified errors and the corresponding troubleshooting measures are documented in detail in APPENDIX D.

### *2.3.2 Custom Protocols and Plugins Developed for the Workflow*

Although Scipion and Scipion-chem offer a wide range of pre-implemented protocols covering most stages of a typical VDS pipeline, during the execution of this study, certain functionalities were found to be missing or insufficient for the specific goals of the workflow.

#### ***2.3.2.1 Protocol for Experimental Binding Energy Integration***

A key limitation was the inability to import experimental binding data from external files into Scipion. To address this, a custom protocol called *Experimental Order* was developed, allowing researchers to enrich ligand sets with experimental affinities and directly benchmark computational predictions, enhancing the VDS workflow's comparative power. The full implementation of the protocol is openly available at https://github.com/veronicagamo/Experimental-Order-Protocol.git

This protocol reads a user-provided CSV (or similarly structured) file that must contain at least two fields: *molName* (the ligand's unique identifier) and the *experimental_energy* (the corresponding experimentally determined binding value). Upon execution, the protocol matches the entries in the file with those in a given *SetOfSmallMolecules* object and appends a new attribute (*experimental_energy*) to the appropriate ligand entries. The user must provide three inputs shown in Figure 19: the path to the CSV file containing the experimental values, the ligand set to be updated, and

the delimiter used in the file (e.g., comma or tab). This last parameter is explicitly requested in the configuration form to follow the design pattern of other Scipion protocols that read data from external files, ensuring consistent file parsing and user experience across the platform.

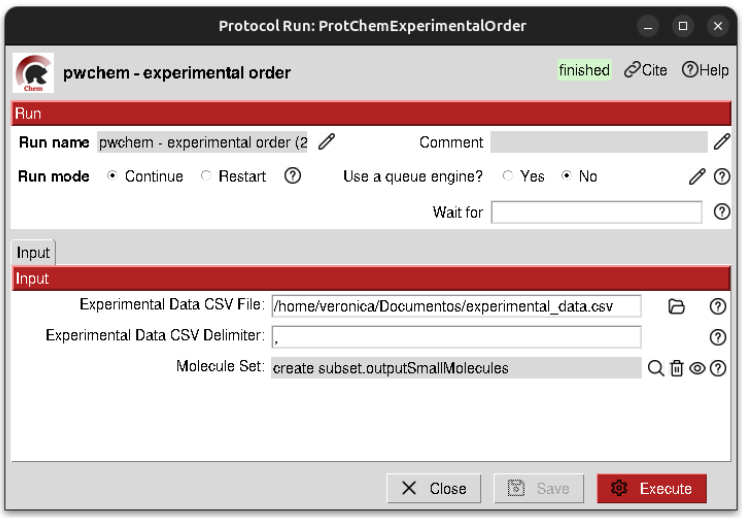



**Figure 19. Protocol Form configuration for the *Experimental Order* custom protocol.**

Internally, the protocol validates the file structure, skips improperly formatted lines and ensures each molecule is correctly annotated without disrupting existing attributes. The resulting output is a new, updated ligand set that can be seamlessly reintegrated into the workflow for downstream tasks, such as, scoring, ranking or statistical correlation with predicted values. This custom protocol is designed to be integrated primarily in the final stages of the VDS workflow (after ranking), as shown in Figure Appendix 6. However, due to its flexible design, the *Experimental Order* protocol can also be incorporated at any point in the pipeline where it is relevant to correlate experimental values with intermediate computational results.

#### *2.3.2.2 Pafnucy Plugin*

To further enhance the predictive power of the VDS workflow, a plugin named Pafnucy was integrated into the Scipion-chem framework. This plugin was developed to simplify and automate the traditionally complex and multi-step process associated with docking workflows (molecular docking, filtering, rescoring, consensus analysis and final

ranking). These operations are not only time-consuming, but also prone to errors. Moreover, within the workflow, they often result in execution failures due to their high computational cost, the large number of ligands typically processed and the complexity of protocol dependencies. Automating and consolidating these steps through the Pafnucy plugin significantly mitigates these issues, enhancing overall workflow stability, efficiency and reproducibility. The plugin adheres to the standard Scipion plugin architecture, as detailed in APPENDIX E, ensuring full compatibility with the broader Scipion ecosystem and facilitating future modular expansions.

The core predictive engine behind this plugin is Pafnucy, a deep learning model originally developed for estimating protein–ligand binding affinities. It was trained on the PDBbind dataset and benchmarked using the CASF scoring power benchmark [41]. Importantly, Pafnucy model remained unaltered; both the neural network architecture and its training process are entirely external to this study. The source code is available at https://github.com/realfolkcode/Pafnucy.git. The contribution here lies in its integration into Scipion, allowing to function seamlessly as part of a larger automated VDS pipeline. The full plugin implementation is openly available at https://github.com/veronicagamo/TFG-Pafnucy-Plugin.git

The Pafnucy model is based on a 3D convolutional neural network that processes voxelized representations of protein–ligand complexes. Each voxel encodes atomic-level features, such as, atom type, hybridization and partial charge. The network architecture consists of three convolutional layers with 64, 128 and 256 filters respectively, each with a filter size of 5 Å, followed by max-pooling layers with a patch size of 2 Å [41]. This is followed by three fully connected (dense) layers with 1000, 500 and 200 neurons [41]. The final layer is a regression output node that predicts the $pK_a$ value (negative logarithm of the acid dissociation constant), which quantifies the binding affinity between the ligand and the protein target. A higher $pK_a$ involves a stronger affinity.

Within Scipion, execution of this model is managed by the *Pafnuncy Ligand-Target Predictions* protocol. It is composed of two main stages. The first stage, *runPreparation*, accepts as input a *SetOfSmallMolecules* object (ligands) and a single *AtomStruct* object (protein structure), as shown in Figure 20. These are combined and

passed to Pafnucy's *prepare.py* script to generate a merged HDF5 input file (*complexes.hdf*). This file encodes the 3D voxelized structures required by the model.

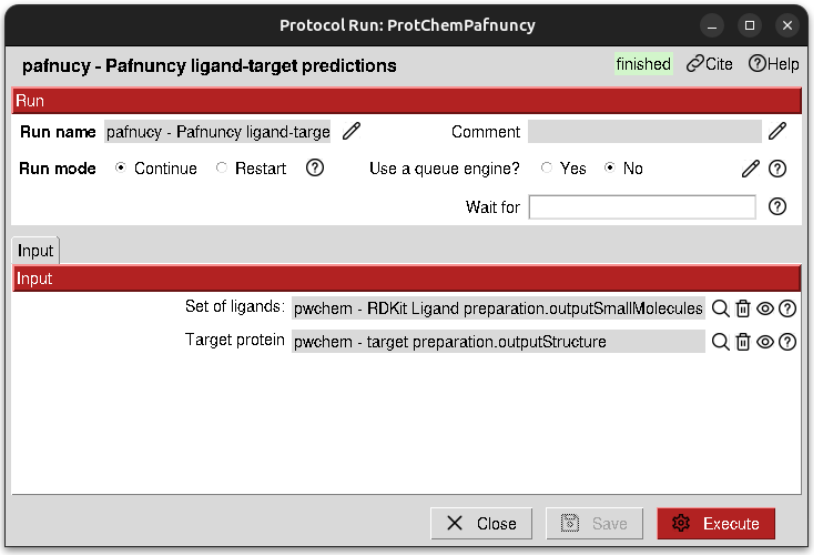



**Figure 20. Protocol Form configuration of the *Pafnuncy Ligand-Target Predictions* protocol.**

The second stage, *runPrediction*, uses Pafnucy's *predict.py* script to process the *complexes.hdf* input and generate a CSV file containing the predicted binding affinities $pK_a$. These predictions are then incorporated into the Scipion environment by appending a new attribute (*Predicted_pKa_Pafnucy*) to each ligand entry. In addition, the associated protein file is stored as metadata for traceability. Thus, the output is an enriched ligand set containing the predicted affinities.

This protocol is designed to be integrated after the ligand and target preparation steps, making it a logical extension once the molecular structures are ready for evaluation. An example of how this protocol fits into the overall workflow is shown in Figure Appendix 7.

#### *2.3.2.3 PLAPT Plugin*

The PLAPT plugin, adapted from an existing tool available at https://github.com/Bindwell/PLAPT.git, was integrated into Scipion-chem to provide deep learning-based binding affinity predictions, enhancing usability, reproducibility and

accessibility within VDS workflows. The full plugin implementation is openly available at https://github.com/veronicagamo/TFG-PLAPT-Plugin.git.

A key advantage of PLAPT is its ability to operate directly on 1D protein sequences, removing the need for 3D structural inputs or computationally expensive preprocessing steps for protein targets, such as protonation or optimization (Figure Appendix 8). This makes it particularly well-suited for early-stage screening or for targets lacking structural data. In this study, protein sequences were imported into Scipion using the *Import Sequence* protocol from Scipion-em plugin, which supports multiple sources including UniProtKB, PDB files and local text entries. ROS1 and RIPK1 sequences were retrieved via UniProtKB using their PubChem IDs [30,31], as shown in Figure 21.

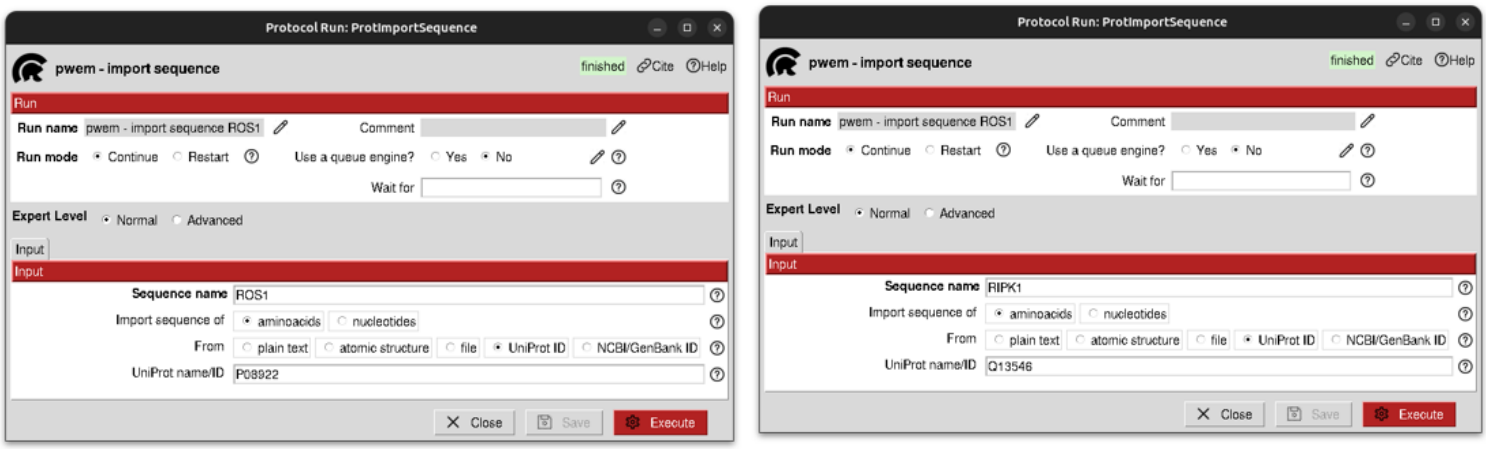



**Figure 21. *Protocol Form* configuration used for importing RIPK1 (right) and ROS1 (left).**

PLAPT leverages transfer learning from pretrained transformers: ProtBERT for proteins and ChemBERTa for ligands [42]. These models generate rich embeddings from sequence data, which are then processed by a branching neural network architecture. Protein and ligand features are first handled by bifurcated linear layers and then concatenated and passed through fully connected layers to output predicted binding affinities as $pK_d$ values. This design enables accurate predictions with minimal computational load, ideal for high-throughput virtual screening.

The plugin is implemented in Scipion through the *PLAPT Analysis* protocol. It takes as input a *SetOfSmallMolecules* object and a *Sequence* object (Figure 22). Ligands are converted to SMILES strings internally and together with the sequence, are fed into PLAPT's prediction engine. Results are returned in JSON format and parsed to extract

the predicted $pK_d$ and corresponding micromolar affinities, which are added as attributes (*pKd_PLAPT*, *Affinity_uM_PLAPT*) to each ligand entry, yielding an updated, analysis-ready ligand set.

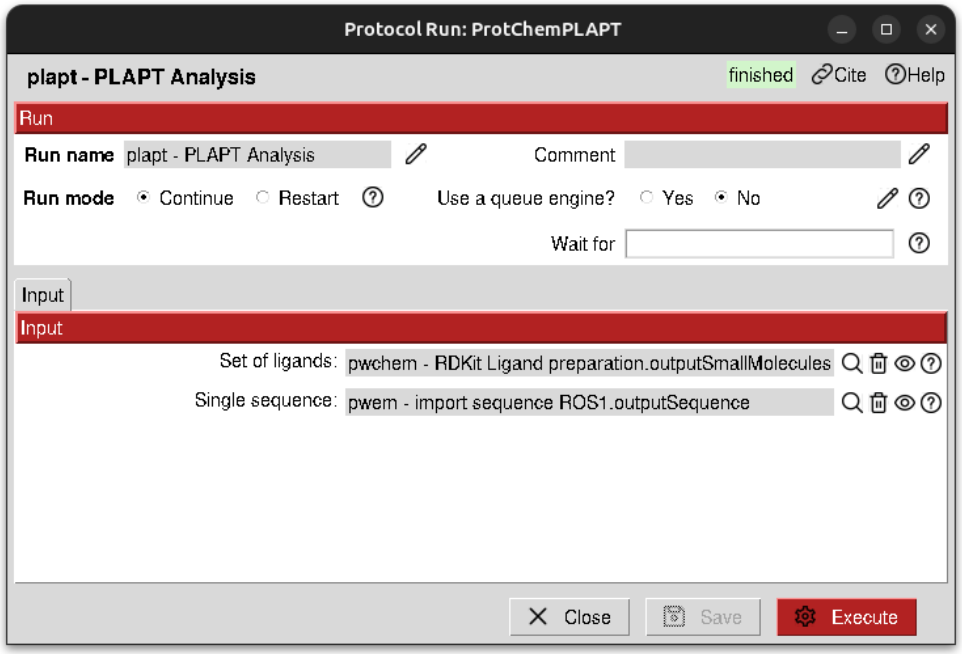



**Figure 22. Configuration for *PLAPT Analysis* protocol.**

To enhance interpretability, the PLAPT plugin includes a dedicated viewer (*ProtChemInteractionViewer*) that converts the model's prediction output into a structured and color-coded HTML report. Unlike other protocols, such as, Pafnucy, which focus solely on numerical predictions, this viewer was specifically designed to offer additional interpretive value by incorporating categorical assessments and visual styling. Upon execution, the report is automatically launched in the user's default browser, enabling quick and intuitive evaluation of prediction results. To access this viewer, users must first click the *Analyze Results* button once the PLAPT protocol has completed. This opens the viewer interface with the available analysis option (Figure 23). From there, clicking on the eye icon launches the interactive HTML report.

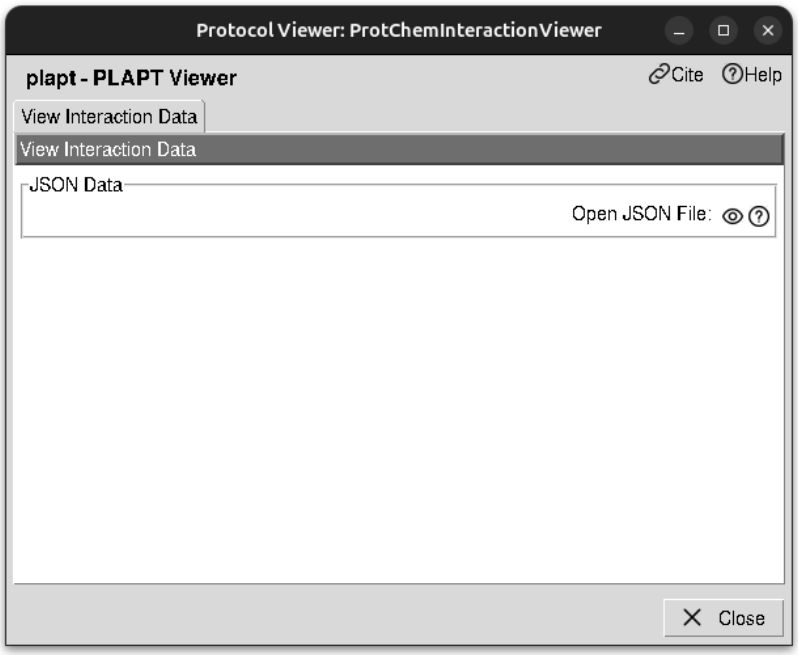



**Figure 23. PLAPT viewer GUI.**

Each ligand entry in the PLAPT output report is displayed in a structured table and includes its SMILES string, the predicted binding affinity (expressed as $pK_d$), the corresponding value in micromolar units (µM) and a qualitative classification of the interaction strength. This classification groups predictions into three categories: high affinity ($pK_d > 8$, highlighted in red), good affinity ($6 \leq pK_d \leq 8$, shown in yellow), and moderate affinity ($pK_d < 6$, displayed in blue). This intuitive color-coding allows users to quickly identify the most promising ligands, facilitating prioritization during high-throughput virtual screening. The classification scheme is based on the interpretative framework proposed by the original PLAPT developers [42].

Overall, the plugin enables rapid and interpretable affinity predictions from sequence inputs and adheres to Scipion's plugin architecture (APPENDIX E), ensuring full integration and future extensibility.

# 3 RESULTS

This section presents the outputs generated at each stage of the VDS workflow for both target proteins: ROS1 and RIPK1. Most visualizations were generated using Scipion's graphical interface and reference-specific figures are available in APPENDIX G when space constraints apply.

Firstly, the evolution of the number of candidate molecules throughout the workflow is shown in Figure 24, including only stages where the compound count was modified. The MD simulation stage is not shown, as it was limited to two ligands per target. Red asterisks in the figure indicate stages where reference ligands were lost. This helps assess how well the workflow retains experimentally validated inhibitors.

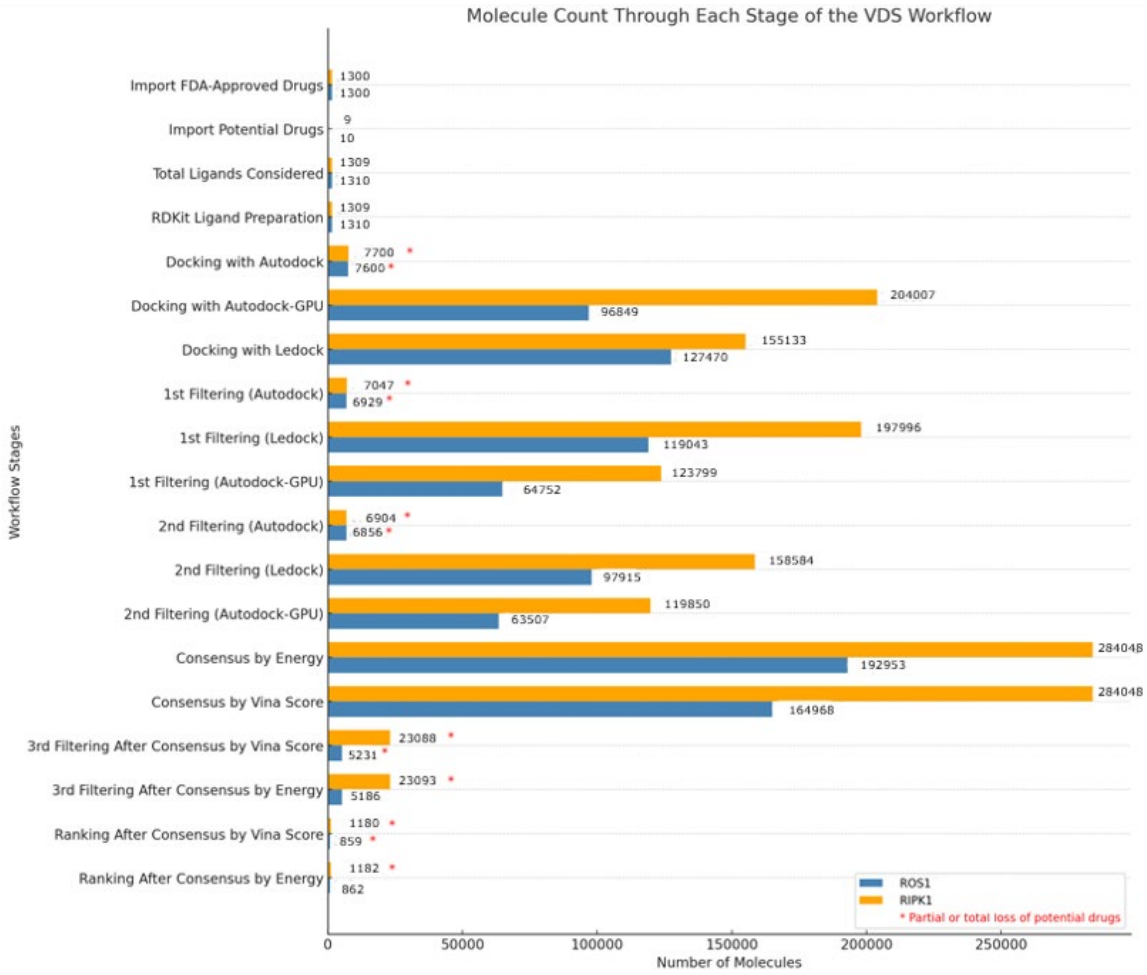



**Figure 24. Evolution of ligand count across the VDS workflow for ROS1 and RIPK1 targets.**

The structures of ROS1 and RIPK1 before and after preparation are displayed in Figure 25 and Figure 26 using PyMOL. Proteins are shown using cartoon and surface representations, with surfaces color-coded according to electrostatic and hydrophobic properties. Prior to preparation, both proteins include co-crystallized ligands and water molecules.

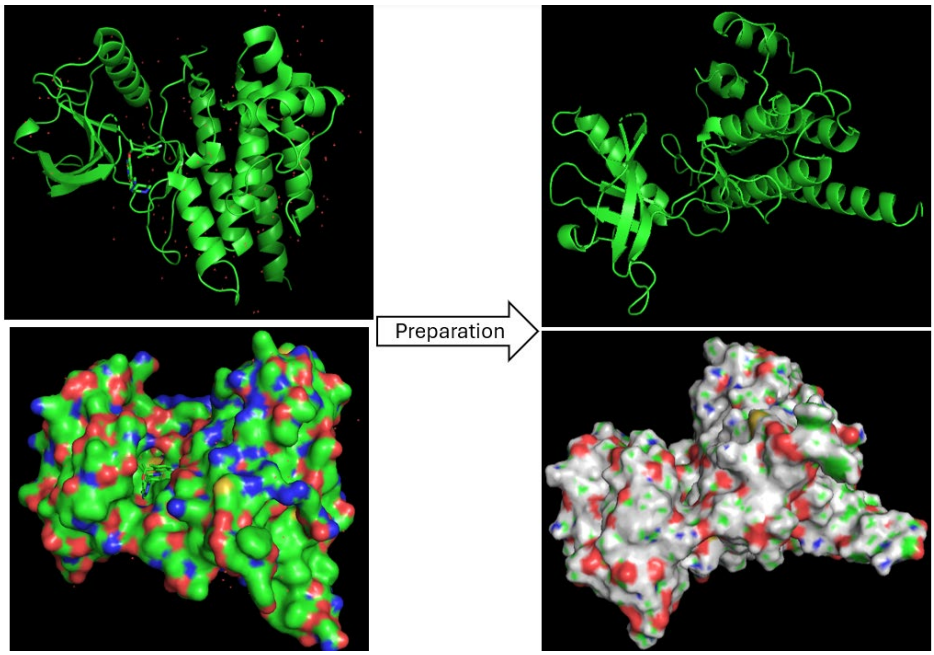



**Figure 25. Structural visualization of ROS1 before and after preparation.**

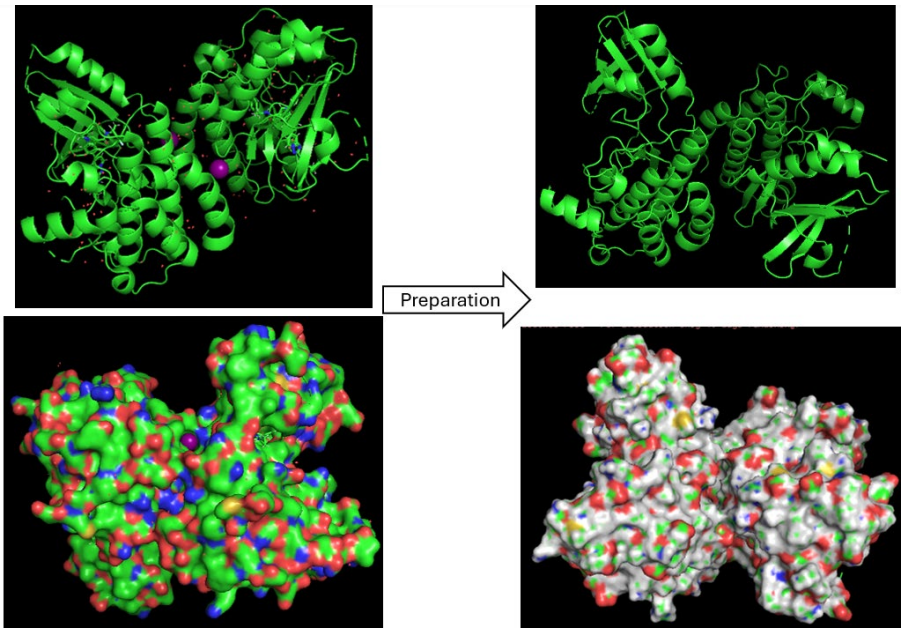



**Figure 26. Structural visualization of RIPK1 before and after preparation.**

Due to space constraints, only protein visualizations are included. Additional examples of ligand structures, before and after preparation, can be found in Figure Appendix 9 and Figure Appendix 10.

Binding site prediction identified 13, 21 and 11 pockets for ROS1 using AutoSite, Fpocket and P2Rank, respectively, consolidated into 10 consensus ROIs. For RIPK1, 26, 34, and 23 pockets were identified, resulting in 16 consensus ROIs. The consensus ROIs are shown in Figure 27.

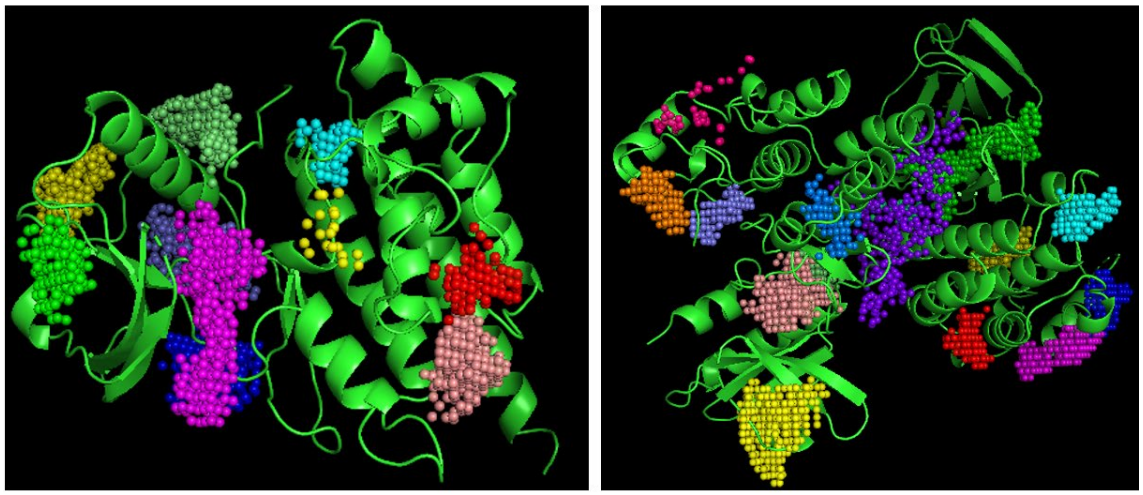


**Figure 27. Consensus ROIs predictions for ROS1 (left) and RIPK1 (right) visualized in PyMOL.**

Docking simulations were run across all consensus-derived pockets. While AutoDock generated poses, it failed to retain any reference inhibitors for ROS1 and preserved only two for RIPK1 (Necrostatin-2 racemate and RIPK1-IN-10). In contrast, LeDock and AutoDock-GPU successfully retained all reference inhibitors. Docking results are stored as tables listing predicted binding energies (*_energy* column) and can also be visualized structurally. A representative table (Figure Appendix 13) and two docked poses (Figure Appendix 14 and Figure Appendix 15) are in APPENDIX G.

Due to the high number of positive and extreme binding energy values — especially in AutoDock with outliers exceeding 700000 kcal/mol, a first filtering step was applied to remove all poses with positive energies. The resulting negative energy distributions for AutoDock are shown in Figure 28, while those for LeDock and AutoDock-GPU are in Figure Appendix 16 and Figure Appendix 17.

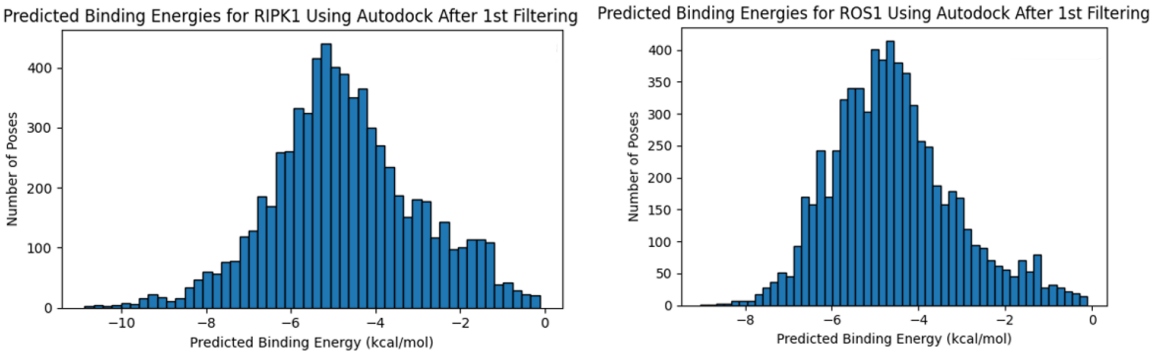



**Figure 28. Distribution of predicted ΔG for RIPK1 and ROS1 after 1st filtering step (AutoDock).**

The first filtering step for LeDock and AutoDock-GPU preserved all referenced inhibitors for both targets.

The next step involved calculating the Vina score using the *ODDT Score Docking* protocol. The output consisted of the same sets of filtered molecules from the previous docking stage, now enriched with an additional attribute in the results table. Since only one scoring function was applied, the new column was labelled *oddtScore1_*. Figure 29 shows scatter plots comparing Vina scores with predicted binding energies for ROS1 and RIPK1 based on AutoDock filtered results. Similar plots for LeDock and AutoDock-GPU are included in Figure Appendix 18 through Figure Appendix 21.

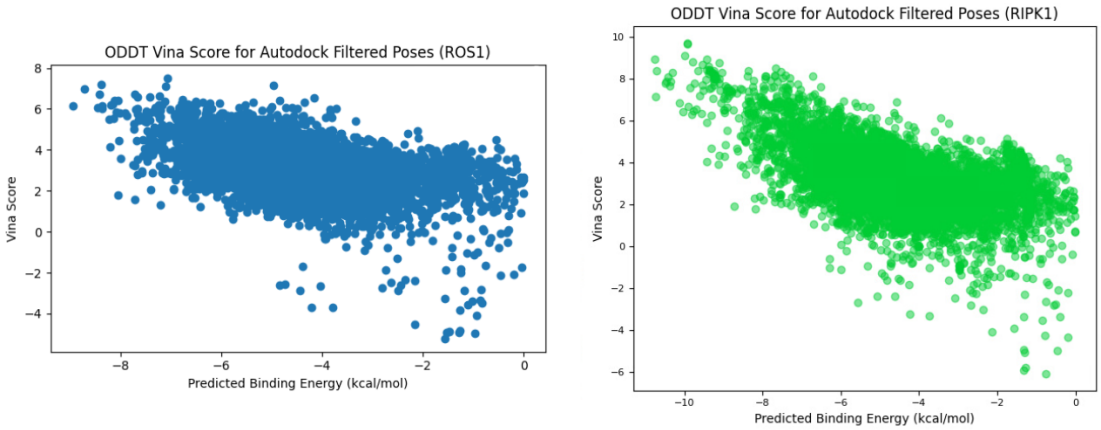



**Figure 29. Scatter plots showing the correlation between AutoDock predicted binding energy and ODDT Vina scores for filtered ROS1 (left) and RIPK1 (right) ligand poses.**

The scatter plots revealed many poses with negative Vina scores, prompting a second filtering step to exclude these unfavourable interactions. After filtering, only favourable poses were retained, and no additional reference inhibitors were lost. The

results after filtering for AutoDock are shown in Figure 30, while those for LeDock and AutoDock-GPU are presented in Figure Appendix 22 through Figure Appendix 25 .

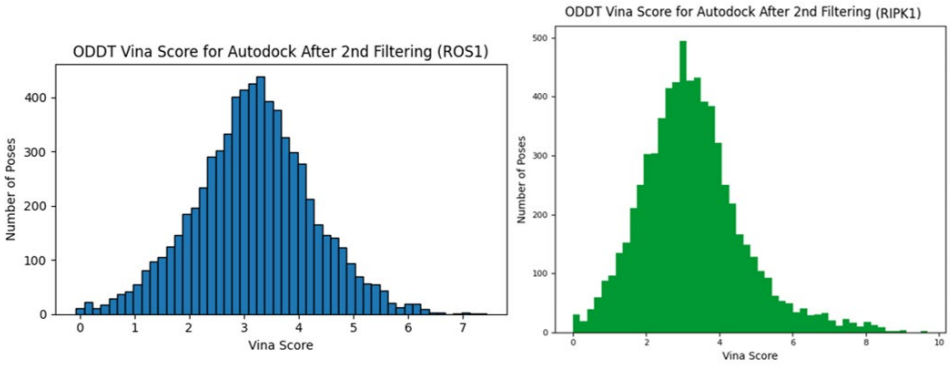



**Figure 30. Vina scores for ROS1 (left) and RIPK1 (right) after second filtering**

After rescoring and filtering, the two consensus protocol configurations were applied (Figure 31). In both cases, no additional reference inhibitors were lost. However, the resulting consensus sets remained too large for downstream processing. Therefore, a third filtering step was applied, retaining only poses with binding energies ≤ –7 kcal/mol. Post-filtering figures are provided in Figure Appendix 26 and Figure Appendix 27.

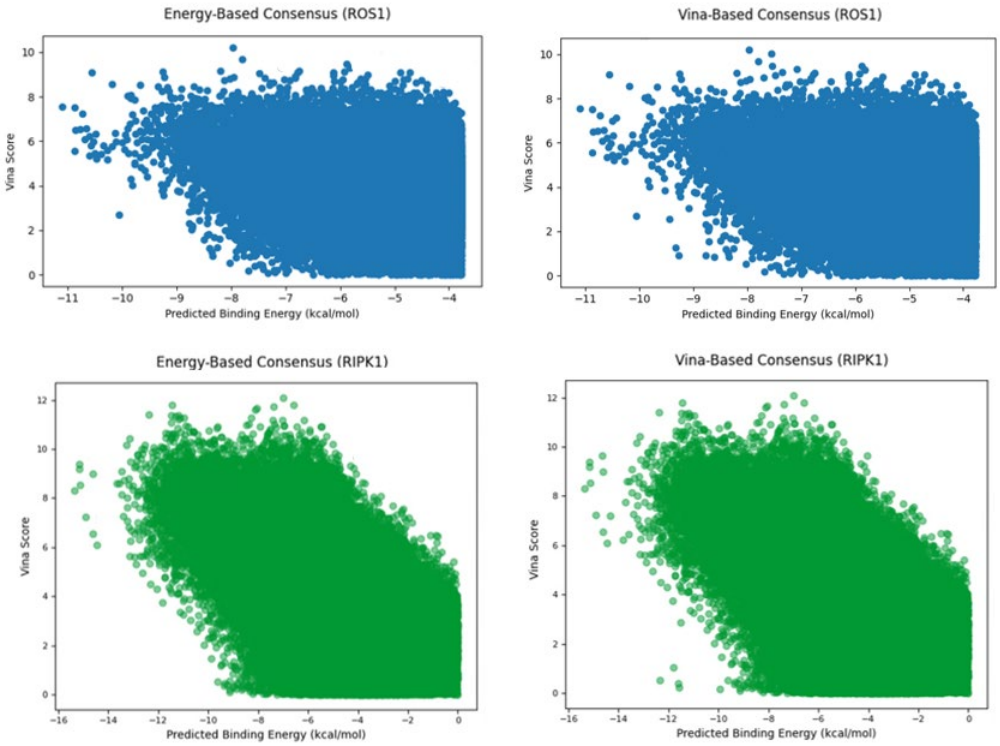



**Figure 31. Consensus results for ROS1 (top) and RIPK1 (down).**

Repotrectinib is absent from the Vina–based consensus set, as all its docking poses exhibited binding energies greater than –7 kcal/mol and were eliminated during the third filtering step. The same exclusion applies to the RIPK1 inhibitors RIPK1-IN-15, RIPK1-IN-10, and 1-[3-(3-Fluorophenyl)-3,4-dihydropyrazol-2-yl]-2,2-dimethylpropan-1-one, regardless of the consensus configuration used.

As in the consensus step, two configurations were tested during the penultimate ranking stage. This protocol generated a new attribute, *rankScore*, for each ligand. The energy, Vina score, and rank score obtained for each reference ligand across the different combinations of consensus and ranking strategies are provided in Table Appendix 1 due to space constraints. Scatter plots displaying the corresponding results for all ligands are included in Figure Appendix 28 through Figure Appendix 35.

Two ligands per target were selected for MD simulations. RMSD overtime is shown in Figure 32 and Figure 33 for ROS1 and RIPK1, respectively. These provide structural validation of binding stability post-docking.

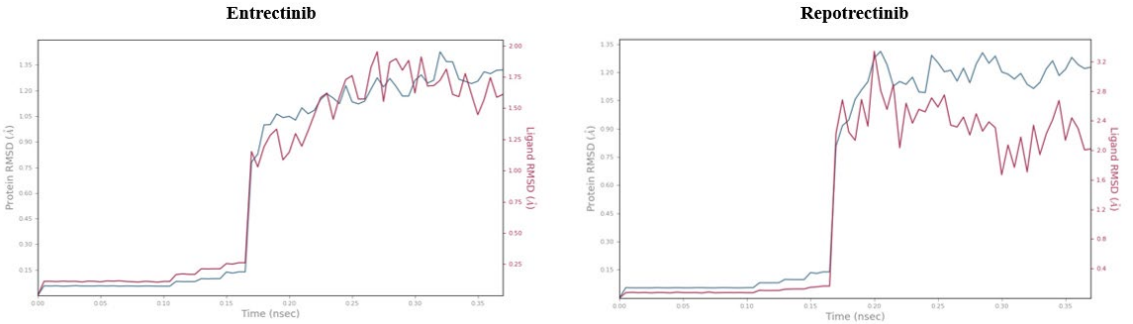



**Figure 32. RMSD trajectories over time for the best and worst ROS1 ligands during MD.**

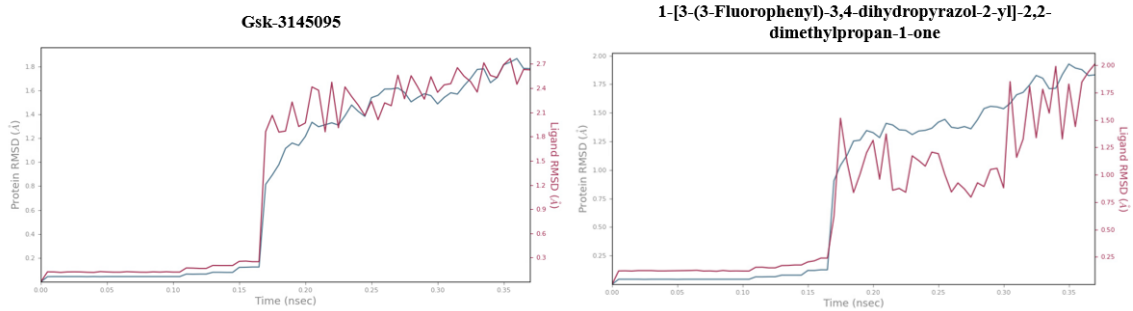



**Figure 33. RMSD trajectories over time for the best and worst RIPK1 ligands during MD.**

The following scatter plots in Figure 34 and Figure 35 display the reference ligands for each protein target, showing the relationship between experimental binding energy and the predicted affinity values: $pK_d$ predicted by the PLAPT model and $pK_a$ predicted by the Pafnucy model. Additionally, Table Appendix 2 includes a table showing the rank positions of the reference ligands among all ligands, sorted in descending order based on their predicted values. This is accompanied by screenshots of the corresponding visual reports generated by the PLAPT viewer interface in Figure Appendix 36 and Figure Appendix 37.

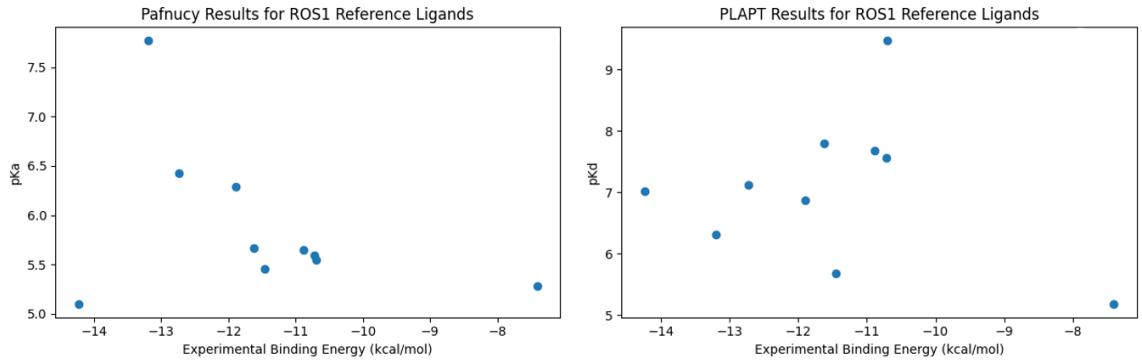



**Figure 34. Predicted affinity vs. experimental ΔG for ROS1 reference ligands using Pafnucy (left) and PLAPT (right).**

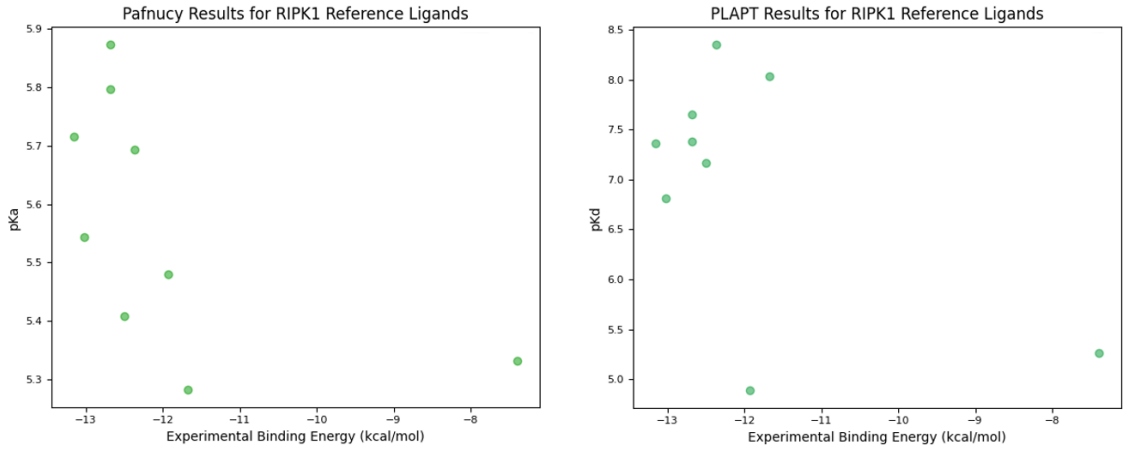



**Figure 35. Predicted affinity vs. experimental ΔG for RIPK1 reference ligands using Pafnucy (left) and PLAPT (right).**

# 4 DISCUSSION

This section critically evaluates the performance of each workflow component, their limitations and the system's overall ability to identify, rank and prioritize ligands with known experimental binding data.

The workflow began with molecular preparation, a critical step for ensuring reliable docking predictions. Ligands prepared using RDKit retained correct three-dimensional geometry and partial charges, whereas OpenBabel produced flattened and chemically unrealistic structures, rendering it unsuitable for this purpose. Protein preparation through the built-in *Target Preparation* protocol effectively cleaned the structures without the need for external tools, as the targets were structurally complete (Figure 25 and Figure 26).

Binding pocket prediction conducted using Fpocket, AutoSite and P2Rank, successfully identified relevant ROIs. Fpocket detected the highest number of pockets and consensus across the tools accurately reproduced known binding sites (Figure 27). For ROS1, the ATP-binding site was consistently identified, while for RIPK1, predicted pockets matched the catalytic cleft and adjacent grooves reported in the literature, confirming the validity of the site detection strategy.

Docking simulations revealed marked differences between engines. AutoDock proved highly unreliable under the present conditions, failing to generate valid poses for a substantial portion of the ligand set, including all reference inhibitors for ROS1 and several for RIPK1. Even in successful cases, AutoDock frequently produced implausible binding energies, often yielding extremely high positive values—far beyond those observed with the other engines—further undermining its reliability. Although LeDock and AutoDock-GPU also occasionally generated poses with positive binding energies, these were comparatively less frequent and never as extreme. Therefore, AutoDock cannot be recommended as a standalone docking tool without additional validation or the support of alternative methods.

Analysis of binding energy distributions after the initial filtering step highlighted notable differences. For ROS1, AutoDock-GPU provided the broadest and deepest energy range (–11,10 to –0,64 kcal/mol), followed by LeDock (–10,61 to 0 kcal/mol), while

AutoDock had the narrowest range (–8,94 to –0,01 kcal/mol). For RIPK1, AutoDock-GPU again generated the most favourable energies (–15,17 to –0,001 kcal/mol), followed by LeDock (–12,83 to 0 kcal/mol) and AutoDock (–10,77 to 0,23 kcal/mol). These results confirm that AutoDock-GPU yielded more thermodynamically favourable conformations, particularly for RIPK1. LeDock also performed well, while AutoDock (Figure 28) showed limited range and failed to recover high-affinity candidates, reinforcing its limitations for robust VDS.

Rescoring with the Vina function (ODDT) showed clear differences in agreement across engines. AutoDock-GPU exhibited the strongest correlation with rescoring results, followed by AutoDock for its valid subset. LeDock presented weaker correlations, especially for RIPK1, suggesting lower alignment between its energy and Vina score. Prior to the second filtering stage, LeDock produced the most negative Vina scores, likely due to geometric optimization not aligned with Vina's scoring criteria. AutoDock-GPU yielded fewer negative scores, indicating better compatibility with rescoring, while AutoDock produced negative scores only in a limited and less reliable set, as illustrated in Figure 29. After removing all negative Vina scores, the retained score ranges for ROS1 were: 0–7,5 (AutoDock), 0–10,21 (LeDock), and 0–9,2 (AutoDock-GPU). For RIPK1, the score ranges were: 0–9,69 (AutoDock), 0–12,09 (LeDock), and 0–11,37 (AutoDock-GPU). While LeDock preserved the widest ranges, AutoDock-GPU offered more uniform and consistent score distributions, supporting its overall robustness in pose quality and rescoring alignment.

A comparative analysis of predicted binding energies, Vina scores and *rankScores* across all consensus and ranking configurations (Table Appendix 1) provided key insights into their ability to replicate experimental affinities. For both ROS1 and RIPK1, energy-based consensus combined with energy-based ranking produced the most negative binding energies and best approximations to experimental ΔG values. Although absolute energies were systematically 2–4 kcal/mol less negative than experimental values, the relative order of ligands was generally preserved. For example, in the case of ROS1, the reference inhibitors Entrectinib, Brigatinib and Ceritinib also exhibited the most favourable predicted binding energies within this configuration.

In terms of ligand prioritization, energy-based consensus combined with Vina-based ranking offered the best separation between strong and weak binders for both target proteins. While *rankScore* values were low, the relative ordering closely matched known binding affinities. For example, Entrectinib and Brigatinib were consistently ranked above weaker compounds, such as, Crizotinib or Repotrectinib. This indicates that Vina rescoring, when applied to reliable docking poses, enhances the prioritization of ligands according to their potency.

By contrast, the configuration using Vina-based consensus with energy-based ranking showed the weakest alignment with experimental data. In some cases, potent reference ligands were ranked below moderate ones—for example, GSK-3145095 received a lower *rankScore* than $C_{24}H_{20}ClN_5O_3$, highlighting potential misprioritization when rescoring is not aligned with thermodynamic strength.

Interestingly, the Vina-based consensus combined with Vina-based ranking yielded results largely comparable to the energy-based consensus with Vina-based ranking for ligands that passed all three filtering stages. However, its more restrictive nature led to the exclusion of several reference ligands during the third filtering step. Specifically, RIPK1-IN-15, RIPK1-IN-10, and 1-[3-(3-Fluorophenyl)-3,4-dihydropyrazol-2-yl]-2,2-dimethylpropan-1-one were eliminated by all configurations at this stage, confirming their consistently poor performance. In contrast, Repotrectinib was excluded exclusively under the Vina-based consensus, while it was retained in energy-based workflows. This reveals a notable trend: when the Vina-based consensus is applied, reference ligands with less negative experimental binding energies—despite their known activity—are more likely to be discarded. These exclusions, although consistent with the applied thermodynamic thresholds, underscore a potential limitation of overly strict consensus strategies, which may filter out moderately active yet biologically relevant compounds.

MD simulations further confirmed the stability of ligand binding modes. As shown in Figure 32 and Figure 33, RMSD trajectories over time clearly distinguish between high- and low-affinity ligands. In ROS1, Entrectinib showed minimal fluctuations (protein and ligand RMSD < 2.0 Å), indicating high structural stability. In contrast, Repotrectinib showed greater variability (ligand RMSD > 2.5 Å), suggesting

weaker binding. Similar trends were observed for RIPK1: GSK-3145095 remained stably anchored in the binding pocket, while low-affinity ligands exhibited significant RMSD fluctuations. These results support a positive correlation between experimental affinity, docking-based prioritization and dynamic stability.

The evaluation of binding affinity predictions for reference ligands using the machine learning models Pafnucy and PLAPT (Figure 34 and Figure 35) revealed notable inconsistencies when compared with experimental binding energies, particularly in the case of ROS1. Theoretically, a higher $pK_d$ and $pK_a$ value should be associated with a more negative experimental binding energy. Additionally, those with the highest pK should correspond to the most potent binders, reflected in top ranking positions. However, this expected relationship was not consistently observed in Table Appendix 2. For ROS1, both models showed a weak and erratic alignment between predicted pK and experimental $\Delta G_{exp}$. Entrectinib, which exhibited the most favourable $\Delta G_{exp}$, was assigned a relatively low pK by both models and ranked near the bottom: 1002nd by Pafnucy and 678th by PLAPT. Conversely, Lorlatinib, with a less favourable $\Delta G_{exp}$, received the highest $pK_d$ value from PLAPT and was ranked 2nd overall, representing a significant overestimation of its binding potential. In contrast, for RIPK1, the models demonstrated better alignment with experimental affinities, particularly in terms of deprioritizing weak ligands. Ligands such as 1-[3-(3-Fluorophenyl)-3,4-dihydropyrazol-2-yl]-2,2-dimethylpropan-1-one, were correctly assigned low pK values and placed in the lower ranks by both models. This suggests that the models reliably identify weak binders, though inconsistencies persist among ligands with moderate to high affinity.

In summary, Pafnucy and PLAPT predictions showed limited agreement with experimental binding energies. Their rank-based prioritization correlated weakly with ligand potency and often misclassified key compounds. Overall, neither model proved reliable for accurate ligand prioritization on its own.

# 5 CONCLUSIONS

This project presents the design and validation of an automated VDS workflow within the Scipion-chem framework, integrating docking, rescoring, consensus analysis, MD simulations and machine learning–based affinity prediction. The system was evaluated using two therapeutically relevant targets, ROS1 and RIPK1, alongside known reference ligands with experimental data and a diverse FDA-approved compound library.

The workflow demonstrated strong predictive capacity, especially under energy-based consensus with Vina rescoring, which consistently preserved the relative ranking of ligands according to their experimental affinities. Among the docking engines tested, AutoDock-GPU yielded the most thermodynamically favourable and consistent results. LeDock also performed reliably, while AutoDock showed significant limitations, frequently failing to generate valid poses or yielding unrealistic energy estimates.

Post-docking scoring and filtering strategies proved effective in distinguishing strong from weak binders. Molecular dynamics simulations further confirmed the structural stability of top-ranked ligands and the instability of poor binders, supporting the robustness of the prioritization strategy. However, machine learning models (Pafnucy and PLAPT) showed variable performance: although useful for deprioritizing weak ligands, they often misranked high-affinity compounds—particularly in ROS1—and thus should be used only in combination with other methods.

In conclusion, the workflow offers a reproducible and modular platform for virtual drug screening, integrating complementary methods to enhance early-stage drug discovery. Future work should focus on incorporating additional docking engines (e.g., Glide, GOLD), improving scoring functions with deep learning and addressing inconsistencies in ML-based affinity prediction through model retraining or contextual enrichment. Moreover, extending the workflow to include ADMET filtering, toxicity prediction, ensemble docking and improved handling of receptor flexibility will increase its pharmacological relevance and scalability for large-scale applications.

# 6 REFERENCES


[1] Paul, S. M., Mytelka, D. S., Dunwiddie, C. T., Persinger, C. C., Munos, B. H., Lindborg, S. R., & Schacht, A. L. (2010). *How to improve R&D productivity: the pharmaceutical industry's grand challenge. Nature Reviews Drug Discovery, 9*(3), 203-214. https://doi.org/10.1038/nrd3078

[2] DiMasi, J. A., Grabowski, H. G., & Hansen, R. W. (2016). *Innovation in the pharmaceutical industry: New estimates of R&D costs. Journal of Health Economics, 47*, 20-33. https://doi.org/10.1016/j.jhealeco.2016.01.012

[3] Lipinski, C. A. (2004). *Lead- and drug-like compounds: The rule-of-five revolution. Drug Discovery Today: Technologies, 1*(4), 337-341. https://doi.org/10.1016/j.ddtec.2004.11.007

[4] Shoichet, B. K. (2004). *Virtual screening of chemical libraries. Nature, 432*(7019), 862-865. https://doi.org/10.1038/nature03197

[5] Pagadala, N. S., Syed, K., & Tuszynski, J. (2017). Software for molecular docking: a review. Biophysical Reviews, 9(2), 91-102. https://doi.org/10.1007/s12551-016-0247-1

[6] Trott, O., & Olson, A. J. (2010). *AutoDock Vina: Improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading. Journal of Computational Chemistry, 31*(2), 455-461. https://doi.org/10.1002/jcc.21334

[7] Durrant, J. D., & McCammon, J. A. (2011). *Molecular dynamics simulations and drug discovery. BMC Biology, 9*(1), 71. https://doi.org/10.1186/1741-7007-9-71

[8] Pagadala, N. S., Syed, K., & Tuszynski, J. (2017). *Software for molecular docking: a review. Biophysical Reviews, 9*(2), 91-102. https://doi.org/10.1007/s12551-016-0247-1

[9] Cherkasov, A., Muratov, E. N., Fourches, D., Varnek, A., Baskin, I. I., Cronin, M., ... & Tropsha, A. (2014). *QSAR modeling: where have you been? Where are you going to? Journal of Medicinal Chemistry, 57*(12), 4977-5010. https://doi.org/10.1021/jm4004285

[10] Schneider, G. (2010). *Virtual screening: An endless staircase? Nature Reviews Drug Discovery, 9*(4), 273-276. https://doi.org/10.1038/nrd3139

[11] Ekins, S., Puhl, A. C., Zorn, K. M., Lane, T. R., Russo, D. P., Klein, J. J., & Hickey, A. J. (2019). *Exploiting machine learning for end-to-end drug discovery and development. Nature Materials, 18*(5), 435-441. https://doi.org/10.1038/s41563-019-0338-z

[12] Scipion. (2022). *Scipion Release 3.0.0 Documentation*. Accessed May 2025, from https://scipion-em.github.io/docs/release-3.0.0/index.html

[13] de la Rosa-Trevín, J. M., Quintana, A., del Cano, L., Zaldívar, A., Foche, I., Gutiérrez, J., Gómez-Blanco, J., Burguet-Castell, J., Cuenca-Alba, J., Abrishami, V., Vargas, J., Otón, J., Sharov, G., Vilas, J. L., Navas, J., Conesa, P., Kazemi, M., Marabini, R., Sorzano, C. O. S., & Carazo, J. M. (2016). Scipion: A software framework toward integration, reproducibility and validation in 3D electron microscopy. *Journal of Structural Biology, 195*(1), 93-99. https://doi.org/10.1016/j.jsb.2016.04.010

[14] Herreros, D., Krieger, J. M., Fonseca, Y., Conesa, P., Harastani, M., Vuillemot, R., Hamitouche, I., Serrano Gutiérrez, R., Gragera, M., Melero, R., Jonic, S., Carazo, J. M., & Sorzano, C. O. S. (2023). Scipion Flexibility Hub: An integrative framework for advanced analysis of conformational heterogeneity in cryoEM. *Acta Crystallographica Section D: Structural Biology, 79*(Pt 7), 569-584. https://doi.org/10.1107/S2059798323004497

[15] Del Hoyo, D., Salinas, M., Lomas, A., Ulzurrun, E., Campillo, N. E., & Sorzano, C. O. (2023). Scipion-Chem: An open platform for virtual drug screening. *Journal of Chemical Information and Modeling, 63*(24), 7873-7885. https://doi.org/10.1021/acs.jcim.3c01085

[16] Scipion-Chem. (2022.). *Scipion-Chem Documentation*. Retrieved May 2025, from https://scipion-chem.github.io/docs/index.html

[17] Weininger, D. (1988). SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules. *Journal of Chemical Information and Computer Sciences, 28*(1), 31-36. http://doi.org/10.1021/ci00057a005

[18] The UniProt Consortium. (2023). UniProt: The Universal Protein Knowledgebase in 2023. *Nucleic Acids Research, 51*(D1), D523–D531. http://doi.org/10.1093/nar/gkac1052

[19] Irwin, J. J., & Shoichet, B. K. (2005). ZINC – A free database of commercially available compounds for virtual screening. *Journal of Chemical Information and Modeling, 45*(1), 177-182. http://doi.org/10.1021/ci049714

[20] Sievers, F., & Higgins, D. G. (2014). Clustal Omega, accurate alignment of very large numbers of sequences. *Methods in Molecular Biology, 1079*, 105-116. http://doi.org/10.1007/978-1-62703-646-7_6

[21] Schlessinger, A., Rost, B., & Honig, B. (2006). Protein structure annotation by local structure prediction. *PLoS Computational Biology, 2*(11), e171. http://doi.org/10.1371/journal.pcbi.0020171

[22] Cock, P. J. A., Antao, T., Chang, J. T., Chapman, B. A., Cox, C. J., Dalke, A., … & de Hoon, M. J. L. (2009). Biopython: Freely available Python tools for computational molecular biology and bioinformatics. *Bioinformatics, 25*(11), 1422-1423. http://doi.org/10.1093/bioinformatics/btp163

[23] Testa, B., & Krämer, S. D. (2008). *The biochemistry of drug metabolism: An introduction*. Wiley. http://doi.org/10.1002/9783527621927

[24] Baell, J. B., & Holloway, G. A. (2010). New substructure filters for removal of pan assay interference compounds (PAINS) from screening libraries and for their exclusion in bioassays. *Journal of Medicinal Chemistry, 53*(7), 2719-2740. http://doi.org/10.1021/jm901137j

[25] Eisenhaber, F., Lijnzaad, P., Argos, P., Sander, C., & Scharf, M. (1995). The double cubic lattice method: Efficient approaches to numerical integration of surface area and volume, and to dot surface contouring of molecular assemblies. *Journal of Computational Chemistry, 16*(3), 273-284. http://doi.org/10.1002/jcc.540160303

[26] Clucas, J., Meier, P. (2023). Roles of RIPK1 as a stress sentinel coordinating cell survival and immunogenic cell death. *Nat Rev Mol Cell Biol*, *24*, 835–852. https://doi.org/10.1038/s41580-023-00623-w

[27] Vandenabeele, P., Declercq, W., Van Herreweghe, F., & Vanden Berghe, T. (2010). The role of the kinases RIP1 and RIP3 in TNF-induced necrosis. *Science Signaling*, 3(115), re4. http://doi.org/10.1126/scisignal.3115re4

[28] Raez, L. E., Manca, P., Rolfo, C., & Singh, V. (2018). *ROS-1 rearrangements in circulating tumor cells*. *Journal of Thoracic Oncology, 13*(5), e71–e72. http://doi.org/10.1016/j.jtho.2017.11.127

[29] Boulanger, M. C., Schneider, J. L., & Lin, J. J. (2024). Advances and future directions in ROS1 fusion-positive lung cancer. *The Oncologist, 29*(11), 943–956. http://doi.org/10.1093/oncolo/oyae205

[30] National Center for Biotechnology Information (2025). PubChem Gene Summary for Gene 8737, RIPK1 - receptor interacting serine/threonine kinase 1 (human). Accessed May, 2025 from https://pubchem.ncbi.nlm.nih.gov/gene/RIPK1/human.

[31] National Center for Biotechnology Information (2025). PubChem Gene Summary for Gene 6098, ROS1 - ROS proto-oncogene 1, receptor tyrosine kinase (human). Accessed May, 2025 from https://pubchem.ncbi.nlm.nih.gov/gene/ROS1/human.

[32] The Science Snail. (2019). *The difference between Ki, Kd, IC50, and EC50 values*. Accessed May, 2025 from https://www.sciencesnail.com/science/the-difference-between-ki-kd-ic50-and-ec50-values

[33] Gilson, M. K., & Zhou, H. X. (2007). Calculation of protein-ligand binding affinities. *Annual Review of Biophysics and Biomolecular Structure*, *36*, 21–42. https://doi.org/10.1146/annurev.biophys.36.040306.132550

[34] Eastman, P., Swails, J., Chodera, J. D., McGibbon, R. T., Zhao, Y., Beauchamp, K. A., Wang, L.-P., Simmonett, A. C., Harrigan, M. P., Stern, C. D., Wiewiora, R. P., Brooks, B. R., & Pande, V. S. (2017). *OpenMM 7: Rapid development of high performance algorithms for molecular dynamics*. PLoS Computational Biology, *13*(7), e1005659. https://doi.org/10.1371/journal.pcbi.1005659

[35] Halgren, T. A. (1999). *MMFF VI. MMFF94s option for energy minimization studies*. Journal of Computational Chemistry, *20*(7), 720–729. https://doi.org/10.1002/(SICI)1096-987X(199905)20:7<720::AID-JCC7>3.0.CO;2-X

[36] Alrouji, M., Yasmin, S., Alhumaydhi, F., Sharaf, S., Shahwan, M., & Shamsi, A. (2024). ROS1 kinase inhibition reimagined: Identifying repurposed drug via virtual screening and molecular dynamics simulations for cancer therapeutics. *Frontiers in Chemistry, 12*. https://doi.org/10.3389/fchem.2024.1392650

[37] Yang, X., Lu, H., Xie, H., Zhang, B., Nie, T., Fan, C., Yang, T., Xu, Y., Su, H., Tang, W., & Zhou, B. (2021). Potent and selective RIPK1 inhibitors targeting dual pockets for the treatment of systemic inflammatory response syndrome and sepsis. *Angewandte Chemie International Edition, 60*(51), 26713–26720. https://doi.org/10.1002/anie.202114922

[38] Alrouji, M., Yasmin, S., Alhumaydhi, F. A., Sharaf, S. E., Shahwan, M., & Shamsi, A. (2024). *ROS1 kinase inhibition reimagined: Identifying repurposed drug via virtual screening and molecular dynamics simulations for cancer therapeutics*. Frontiers in Chemistry, *12*. https://doi.org/10.3389/fchem.2024.1392650

[39] Zhao, H., & Caflisch, A. (2013). Discovery of ZAP70 inhibitors by high-throughput docking into a conformation of its kinase domain generated by molecular dynamics. *Bioorganic & Medicinal Chemistry Letters*, *23*(21), 5721–5726. https://doi.org/10.1016/j.bmcl.2013.08.094

[40] Poli, G., Martinelli, A., & Tuccinardi, T. (2016). Reliability analysis and optimization of the consensus docking approach for the development of virtual screening studies. *Journal of Enzyme Inhibition and Medicinal Chemistry*, *31*(sup2), 167–173. https://doi.org/10.1080/14756366.2016.1193736

[41] Stepniewska-Dziubinska, M. M., Zielenkiewicz, P., & Siedlecki, P. (2018). Development and evaluation of a deep learning model for protein–ligand binding affinity prediction. *Bioinformatics, 34*(21), 3666–3674. https://doi.org/10.1093/bioinformatics/bty374

[42] Rose, T., Monti, N., Anand, N., & Shen, T. (2024). *PLAPT: Protein-ligand binding affinity prediction using pretrained transformers*. bioRxiv. https://doi.org/10.1101/2024.02.08.575577

# APPENDIX

## *A. Scipion Installation*

If you don't have conda installed, check by running the following command in your console:

```
which conda
```

If conda is not installed, follow the steps below to install Miniconda. Alternatively, proceed to step 3.

1. Install Miniconda:

```
wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh
```

```
bash Miniconda3-latest-Linux-x86_64.sh -b -p /path/to/miniconda
```

2. Verify you are using bash:

```
echo $SHELL
```

3. Initialize conda:

```
source /path/to/miniconda/etc/profile.d/conda.sh
```

4. Activate the base conda environment and install the Scipion installer:

```
conda activate
```

```
pip3 install --user scipion-installer
```

5. Install Scipion and generate default config files:

```
python3 -m scipioninstaller -conda -noAsk /path/to/scipion
```

```
/path/to/scipion/scipion3 config –overwrite
```

6. To make launching Scipion easier, add the following alias to your *.bashrc* file:

```
alias scipion3='/path/to/scipion/scipion3'
```

To launch Scipion, you only need to type the name of the alias you selected in your *.bashrc* file in the terminal.

## *B. Plugin Installation*

To install each plugin, follow these steps:

1. Clone the plugin repository (replace <plugin-repo-url> with the actual plugin repository URL):

```
git clone <plugin-repo-url>
```

2. Navigate to the cloned directory:

```
cd <plugin-directory>
```

3. Checkout the development branch (if necessary):

```
git checkout devel
```

4. Install the plugin with Scipion (replace <path-to-plugin> with the path to the cloned plugin directory):

```
scipion3 installp -p <path-to-plugin> --devel
```

For example, to install the Scipion-chem plugin, follow these steps:

```
git clone https://github.com/scipion-chem/scipion-chem.git
```

```
cd scipion-chem
```

```
git checkout devel
```

```
scipion3 installp -p /path/to/scipion-chem –devel
```

## *C. VDS Workflow Overview*

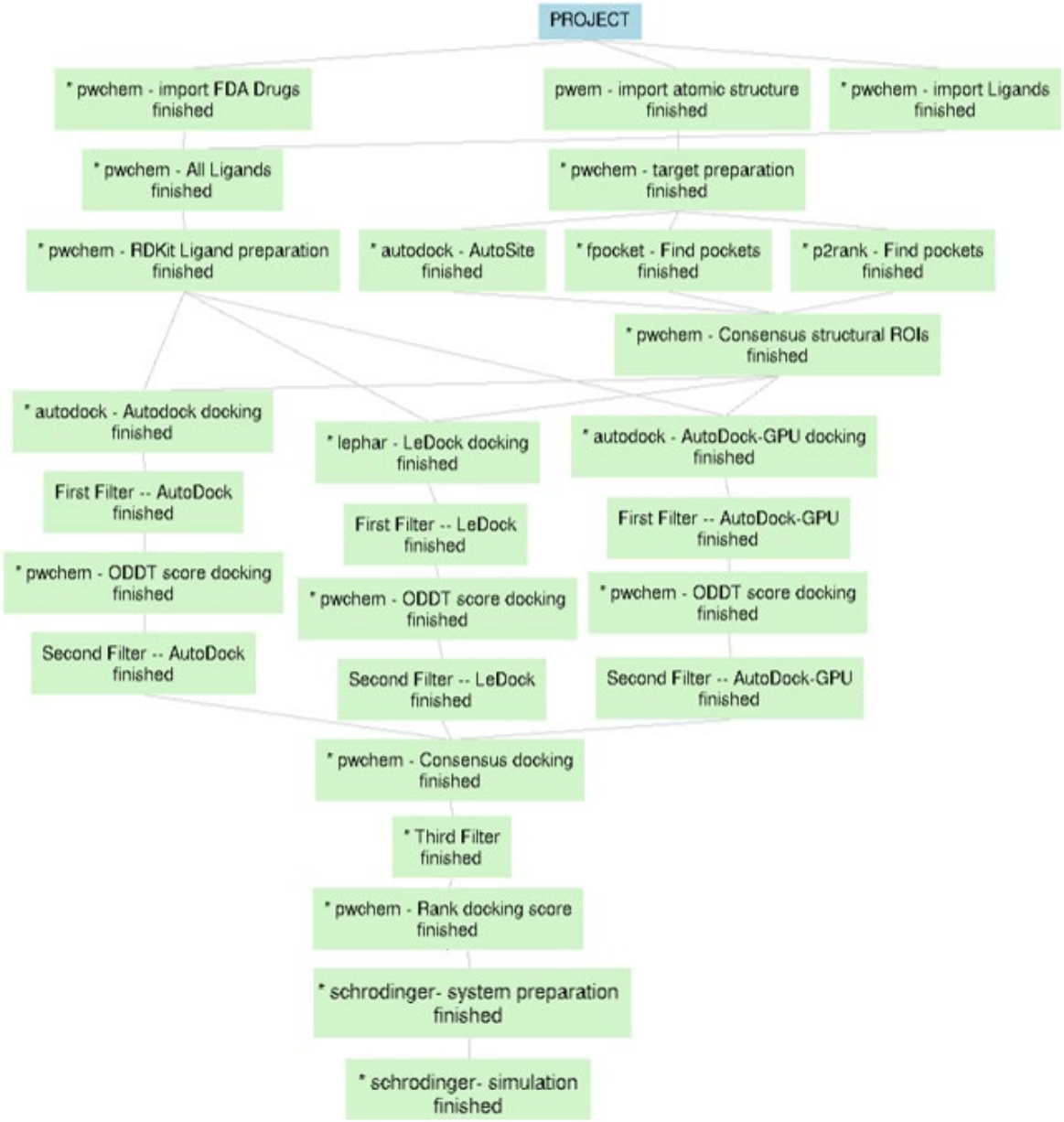



**Figure Appendix 1. Visual representation of the VDS workflow in Scipion.**

## *D. Issues and Challenges Encountered*

Several protocols encountered runtime failures caused by software bugs or hardware limitations. Memory allocation issues and extended execution times became significant obstacles during computationally intensive stages, such as large-scale docking and consensus analysis. These phases, particularly when processing large ligand libraries, occasionally lead to workflow interruptions or system crashes. To mitigate these

challenges, it was often necessary to manually adjust protocol parameters, split tasks into smaller batches and re-run failed steps.

Crucially, all technical issues identified during the implementation process were carefully communicated to the Scipion-chem development team. In some instances, proposed solutions (direct code modifications) were suggested to resolve the problems. These contributions were reviewed by the developers, who evaluated their suitability and decided whether to incorporate them into the official version of the plugin based on their technical soundness and overall utility.

## *D.1 Conflict with Object IDs in Rank Docking Protocol*

During the execution of the *Rank Docking Score* protocol, a runtime error was encountered due to ID conflicts in the internal database. The error traceback revealed a `sqlite3.IntegrityError: UNIQUE constraint failed: Objects.id`, which indicated that newly generated molecules were being assigned duplicate object identifiers.

To resolve the issue, a manual fix was introduced at line 147 of the protocol script (*protocol_rank_docking_score.py*) from https://github.com/scipion-chem/scipion-chem.git by resetting the object ID using:

```
mol.setObjId(None)
```

This command clears the current object ID, allowing Scipion's SQLite-based backend to automatically assign a new, unique ID upon insertion. Importantly, this operation only affects the ID of the current molecule object and does not modify other entries in the dataset. It is unknown whether a permanent fix has been applied in more recent versions of the Scipion-chem plugin.

## *D.2 Conflict with Parameter Parsing in OpenMM's System Preparation*

Although Schrödinger's Desmond engine was ultimately chosen for the final simulations, other MD tools, such as, OpenMM were also explored during the evaluation phase.

An error was encountered during the execution of the *OpenMM System Preparation* protocol when attempting to read the *solvationParams.txt* file. The issue was traced to the original parameter parsing function. The specific traceback observed was:

```
Traceback (most recent call last):

File "/home/veronica/scipion-chem
openmm/openmm/scripts/openmmPrepareSystem.py", line 69, in <module> sysName =
os.path.splitext(os.path.basename(pDic['receptorFile']))[0]

KeyError: 'receptorFile'
```

This error occurred because the original function, *parseParams()*, did not account for missing or incorrectly formatted lines in the parameter file. To resolve this, a custom version of the *parseParams()* function was implemented in the *utils.py* script in the Scipion-chem-openmm plugin (available at https://github.com/scipion-chem/scipion-chem-openmm.git).

The revised function introduced several key enhancements. It handles malformed lines that do not contain the expected separator (::) safely, preventing unexpected crashes during execution. Additionally, it avoids exceptions by skipping improperly formatted entries rather than attempting to process them. Finally, it ensures clean and consistent data parsing by removing unnecessary whitespace from both keys and values. The custom implementation is shown below:

```
def parseParams(filename, sep='::'):
    pDic = {}
    with open(filename, 'r') as f:
        for line in f:
            line = line.strip()
            if sep in line:
                key, value = line.split(sep, 1)
    return pDic
```

### *D.3 Obsolete Performance of OpenBabel in Ligand Preparation*

Another issue encountered during the workflow was related to ligand preparation using the *OpenBabel Ligand Preparation* protocol. Although the protocol executed without any reported errors, the resulting molecular structures were completely planar, indicating a failure in generating proper 3D conformers. This issue affected not only ligands manually downloaded from PubChem, but also those imported directly from the ZINC database, demonstrating that the problem was systematic rather than input specific.

The underlying cause appears to be the outdated state of the OpenBabel integration within Scipion-chem. As this tool is no longer actively maintained in the platform, its reliability and compatibility with current molecular formats have significantly declined. Moreover, its lack of error feedback allowed incorrect geometries to pass unnoticed, further complicating downstream analyses.

Due to these limitations, RDKit was adopted as the preferred tool for ligand preparation. Its modern architecture, active development and robust geometry handling ensured consistent and chemically accurate structures across all ligand sources, making it a more dependable choice for high-throughput virtual screening workflows. Figure Appendix 2 and Figure Appendix 3 illustrate two examples of ligand structures visualized in PyMOL visualization tool. On the left, the molecules 1-[3-(3-Fluorophenyl)-3,4-dihydropyrazol-2-yl]-2,2-dimethylpropan-1-one (RIPK1 inhibitor) and ZINC000000000456 are shown after being processed with the OpenBabel, exhibiting fully planar geometries. On the right, the same molecules are shown after preparation with RDKit, displaying proper three-dimensional geometry and realistic molecular conformations.

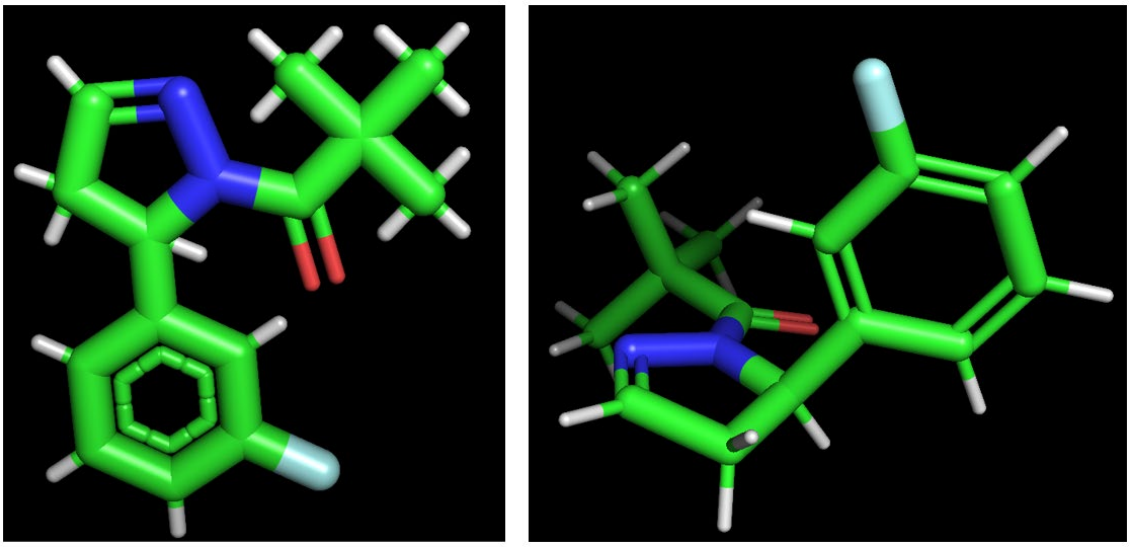


**Figure Appendix 2. Distorted planar geometry of RIPK1 inhibitor after OpenBabel preparation (left) compared to correct 3D structure via RDKit (right).**

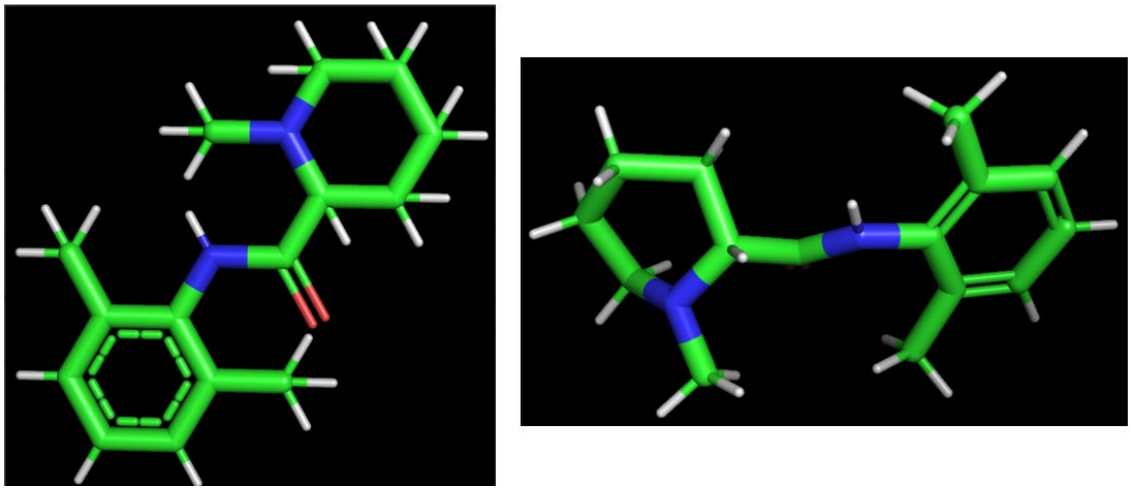


**Figure Appendix 3. ZINC ligand rendered as fully planar by OpenBabel (left) versus properly prepared 3D conformer using RDKit (right).**

## *D.4 Addressing Resource Bottlenecks in Large-Scale Docking*

Finally, another significant challenge encountered during the execution of the workflow was the limitation of computational resources. Several protocols, particularly those involving molecular docking and MD, either failed to complete or required prohibitively long runtimes. These performance issues were primarily due to the large number of ligands included in the screening phase, the presence of multiple predicted

binding pockets per target and the inherently high computational demands of MD simulations.

To address these bottlenecks, a practical optimization strategy was implemented during the docking stage. After ligand preparation, the full dataset was divided into smaller subsets, typically containing around 500 molecules each. This subdivision allowed docking protocols to be executed independently for each subset, significantly reducing memory consumption and minimizing execution failures.

Subset generation was carried out using the *Split Set* protocol from Scipion-em (Figure Appendix 4), which allows users to divide a ligand set into a user-defined number of parts. The number of partitions can be adapted to the size of the dataset and an optional randomization feature ensures an even distribution of compounds across subsets.

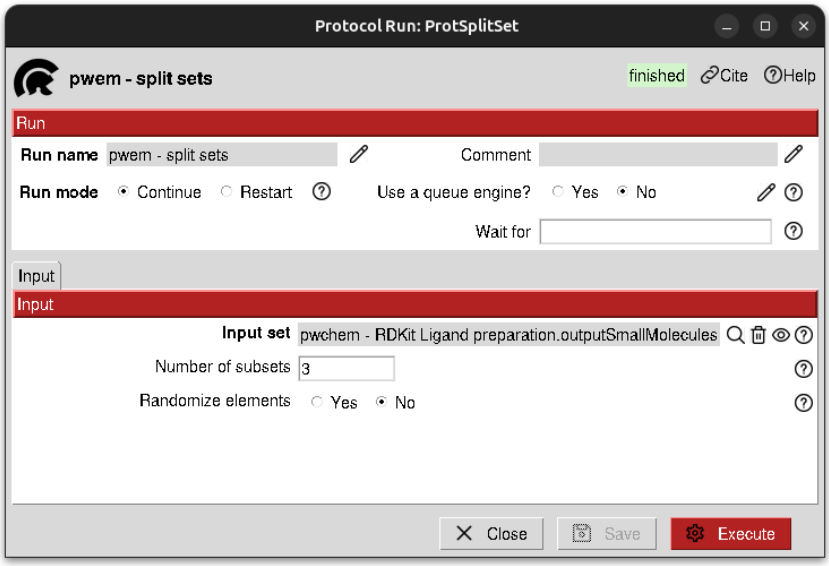



**Figure Appendix 4. Configuration of the *Split Set* protocol.**

Once docking was completed for all subsets, the results were recombined using the *Operate Set* protocol in *Union* mode, enabling the workflow to proceed without interruption. Although splitting is not strictly mandatory, it is strongly recommended when working with large ligand libraries, as it improves both performance and reliability. A visual overview of this procedure is presented in Figure Appendix 5, which summarizes the subset generation workflow and its integration within the docking phase of the VDS pipeline.

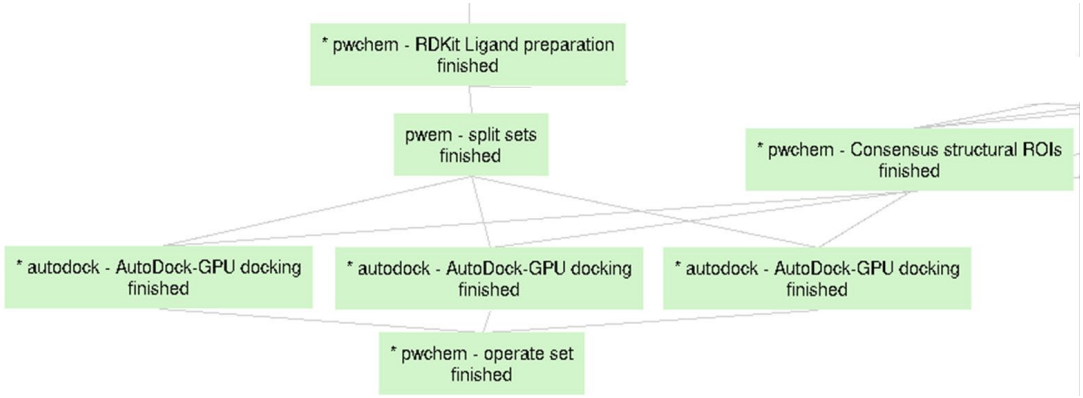



**Figure Appendix 5. Integration of the *Split Set* protocol into the VDS workflow.**

For example, the AutoDock-GPU protocol encountered runtime issues when attempting to dock all ligands simultaneously. In the case of the ROS1 target, executing the protocol without subset partitioning required over 11 hours. In contrast, for RIPK1, where ligands were divided into three subsets, each subset completed in less than 3 hours. This clearly demonstrates how partitioning enhances workflow scalability when tackling resource-intensive tasks.

## *E. Plugin Structure Adopted for Integrated Tools*

The design and implementation of the plugins developed in this project followed the standard Scipion plugin architecture. This modular structure ensures compatibility with the Scipion framework and allows for easy development, integration, and maintenance of new functionalities within the platform.

Each plugin is implemented as a Python package, with a clearly defined folder hierarchy and configuration files. The directory structure adopted for the plugins in this work follows this canonical layout:

scipion-chem-namePlugin/

├── CHANGES.txt

├── LICENSE

├── MANIFEST.in

├── README.rst

```
├── requirements.txt
├── setup.py
├── myplugin/
│   ├── __init__.py
│   ├── bibtex.py
│   ├── constants.py
│   ├── objects.py
│   ├── protocols.conf
│   ├── protocols/
│   │   ├── __init__.py
│   │   ├── protocol_namePlugin.py
│   ├── viewers/
│   │   ├── __init__.py
│   │   ├── viewer_nameProtocol.py
│   ├── wizards/
│   │   ├── __init__.py
│   │   ├── wizard_nameProtocol.py
│   ├── tests/
│   │   ├── __init__.py
│   │   ├── test_nameProtocol.py
```

At the root level, several files handle metadata, configuration, and distribution. *CHANGES.txt* documents the plugin's version history and update log. *LICENSE* outlines the licensing terms under which the plugin is distributed. *MANIFEST.in* defines which non-code files (e.g., documentation or icons) should be included during packaging. *README.rst* contains a long-form description of the plugin, useful for users and for

publishing the plugin to PyPI. *Requirements.txt* lists Python dependencies that the plugin needs. Lastly, *setup.py* specifies how the plugin is installed, including its metadata and where it appears in Scipion's protocol tree.

The core plugin logic resides in the *myplugin/* directory. Here, *__init__.py* defines the main Plugin class and sets up its environment. *Bibtex.py* includes BibTeX-format references that Scipion uses to automatically manage citations. *Constants.py* stores global constants such as environment variable names. *Objects.py* defines custom Python objects or data structures used within protocols. The *protocols.conf* file configures how the plugin's protocols are organized and displayed in Scipion's graphical interface.

The *protocols/* subdirectory contains the implementation of the plugin's computational logic. It includes its own *__init__.py* and one or more protocol definition files, such as *protocol_namePlugin.py*, which encapsulate the core functionality.

Two additional subdirectories, *viewers/* and *wizards/*, are optional but can be included when needed. The *viewers/* folder enables custom graphical representations of results, enhancing the interpretation of complex outputs. The *wizards/* folder provides interactive configuration tools that help users set parameters correctly through guided interfaces. If such custom visualization or configuration tools are not required, these folders may remain empty.

Lastly, the *tests/* directory contains testing scripts to validate the correct operation of the plugin. It includes *__init__.py* and files such as *test_nameProtocol.py*, which are used during development and deployment to verify that the protocols function as expected within the Scipion environment.

## F. Overview of the Workflows Integrating Custom-Developed Protocols

Figure Appendix 6, Figure Appendix 7 and Figure Appendix 8 illustrate the sequential organization of the custom protocols developed in this project, integrated into a structured and automated pipeline within the Scipion-chem framework.

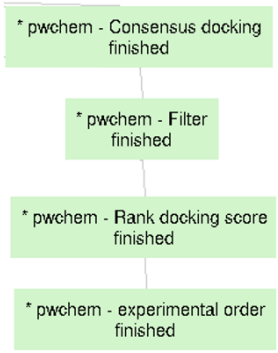



**Figure Appendix 6. Integration point of the *Experimental Order* protocol within the workflow.**

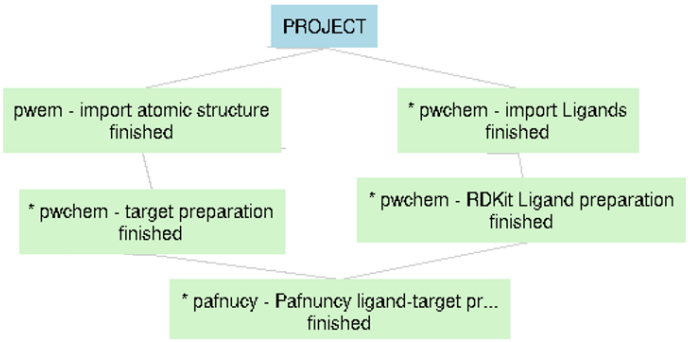



**Figure Appendix 7. Workflow overview integrating Pafnucy.**

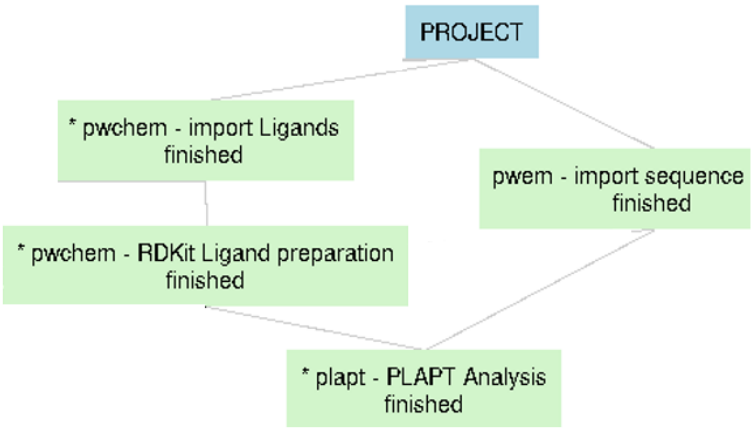



**Figure Appendix 8. Workflow overview integrating PLAPT.**

## *G. Extended Results*

This section contains supplementary figures corresponding to the RESULTS section that were not included in the main body of the document due to space constraints.

These additional visualizations support and complement the analyses presented throughout the workflow.

Figure Appendix 9 and Figure Appendix 10 illustrate two ligands—Entrectinib, a potential inhibitor of ROS1, and ZINC000410428674—visualized in PyMOL before and after structural preparation. These images highlight key changes introduced during the preparation process, such as geometry optimization and hydrogen atom addition. As observed in the case of Entrectinib, no significant modifications were necessary, since the structure, like all ligands imported from PubChem, was already downloaded in a pre-optimized 3D format. In contrast, the ligand retrieved from the ZINC database, representative of all ZINC-imported compounds, underwent notable structural adjustments during preparation to ensure proper 3D conformation and molecular integrity.

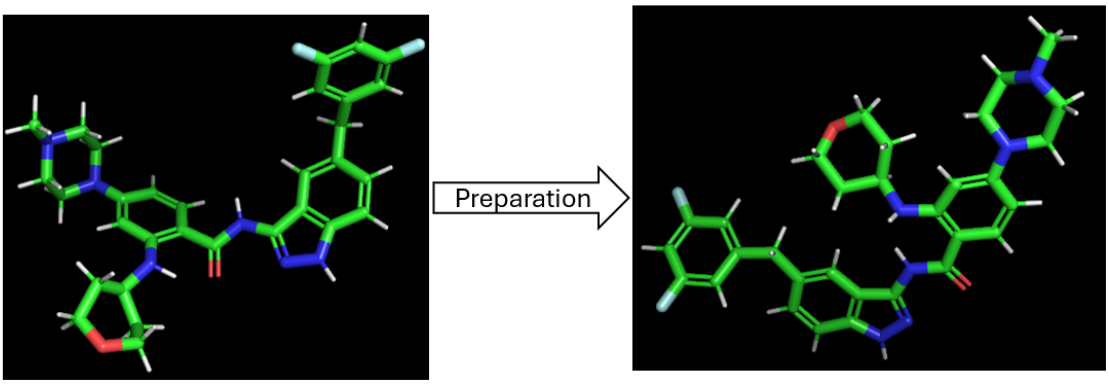



**Figure Appendix 9. Structural visualization of Entrectinib before and after preparation.**

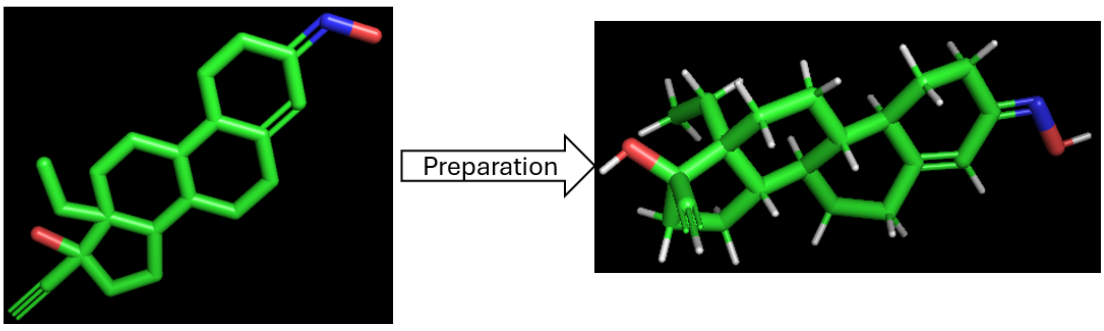



**Figure Appendix 10. Structural visualization of ZINC000410428674 before and after preparation.**

Additional visualizations of individual ROI predictions for ROS1 and RIPK1, generated by AutoSite, Fpocket and P2Rank, are provided in Figure Appendix 11 and Figure Appendix 12. They include full surface renderings and pocket annotations to support the consensus analysis presented in the main text.

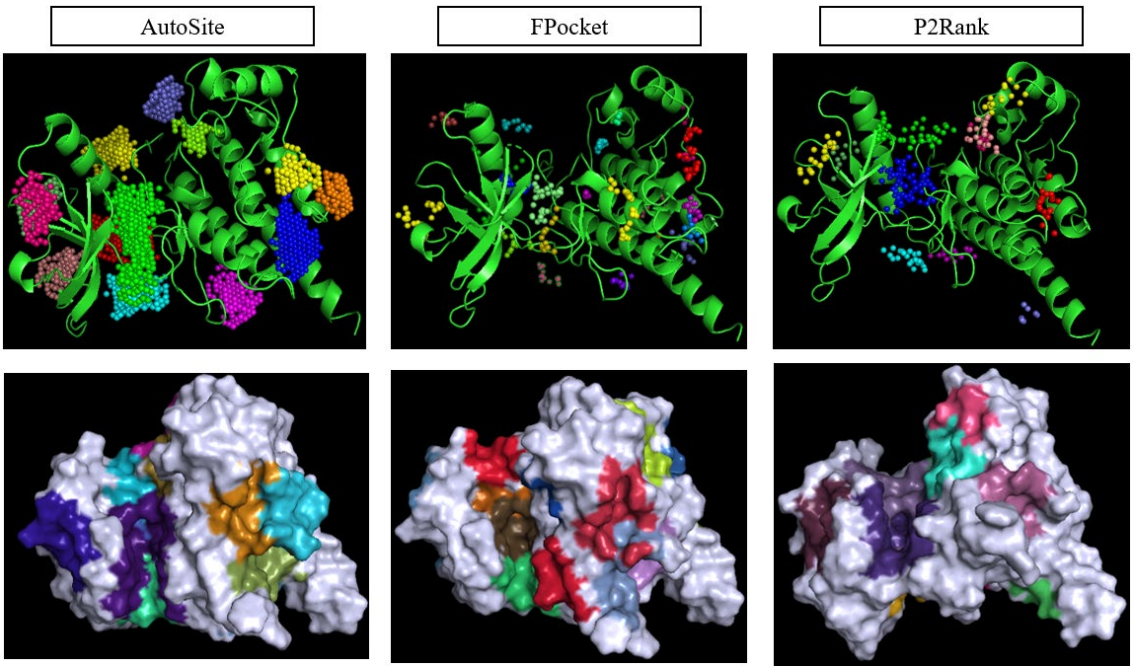



**Figure Appendix 11. ROI predictions for ROS1 by AutoSite, Fpocket and P2Rank.**

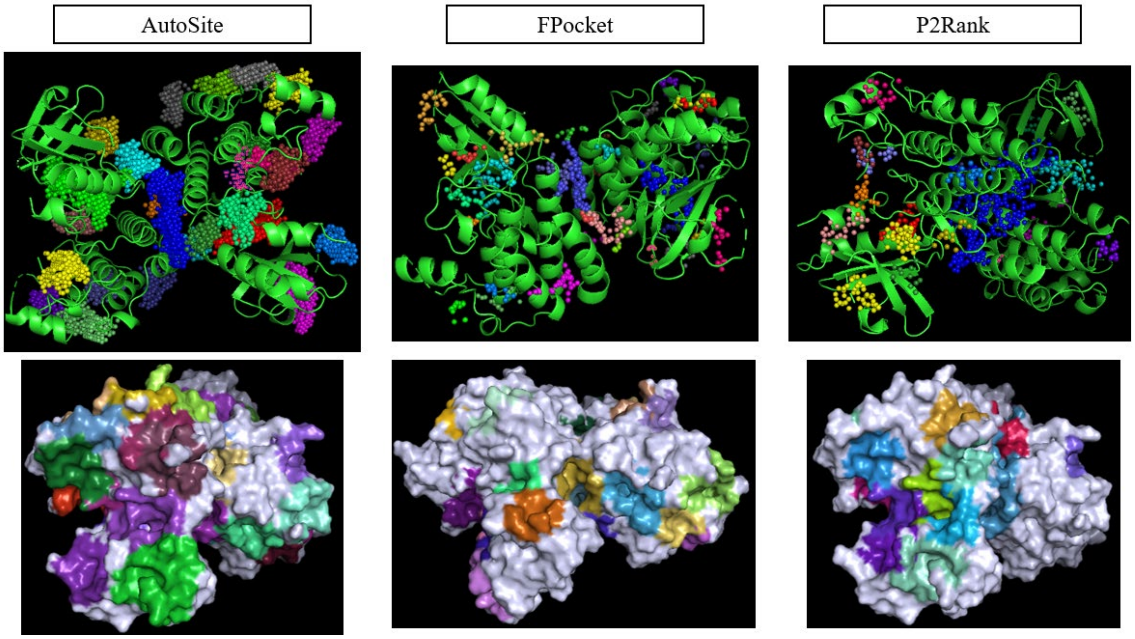
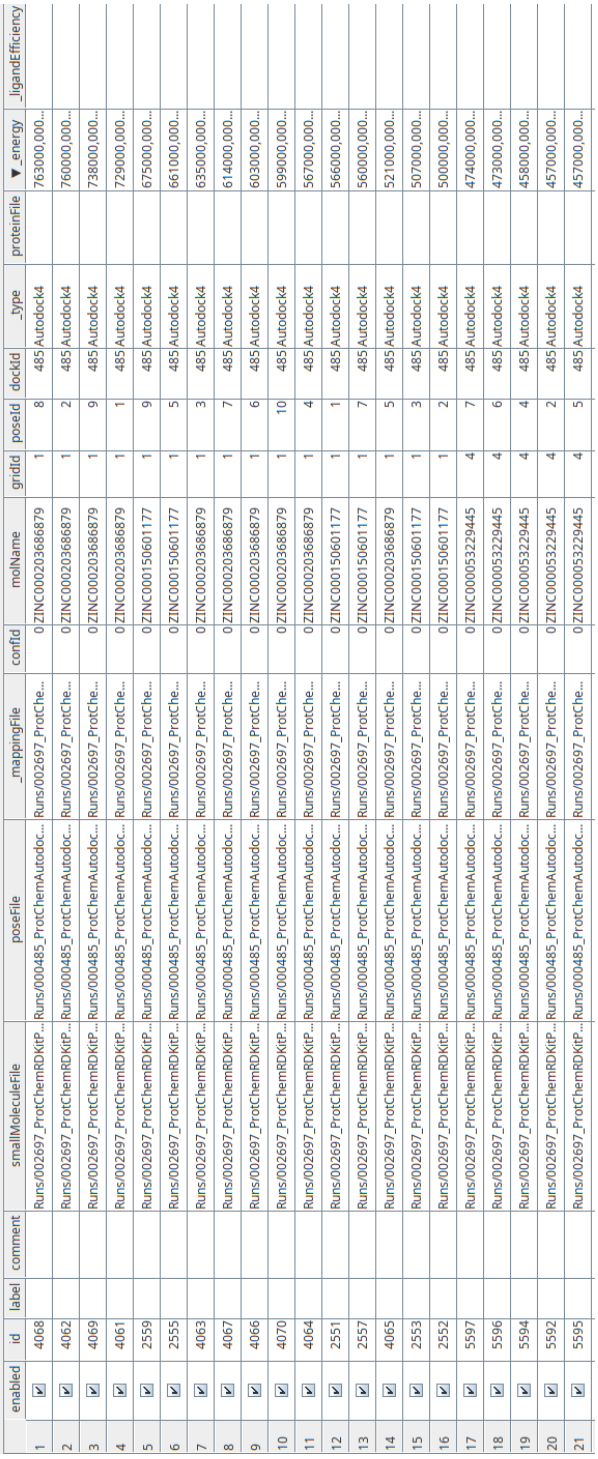



**Figure Appendix 12. ROI predictions for RIPK1 by AutoSite, Fpocket and P2Rank.**

The table in Figure Appendix 13 provides an example of the output format generated by any of the docking protocols used in the workflow, including the predicted binding energy for each ligand–pocket complex.



| | enabled | id | label | comment | smallMoleculeFile | poseFile | _mappingFile | confId | molName | gridId | poseId | dockId | _type | proteinFile | ▼_energy | _ligandEfficiency |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | ☑ | 4068 | | | Runs/002697_ProtChemRDKitP... | Runs/000485_ProtChemAutodoc... | Runs/002697_ProtChe... | 0 | ZINC000203686879 | 1 | 8 | 485 | Autodock4 | | 763000,000... | |
| 2 | ☑ | 4062 | | | Runs/002697_ProtChemRDKitP... | Runs/000485_ProtChemAutodoc... | Runs/002697_ProtChe... | 0 | ZINC000203686879 | 1 | 2 | 485 | Autodock4 | | 760000,000... | |
| 3 | ☑ | 4069 | | | Runs/002697_ProtChemRDKitP... | Runs/000485_ProtChemAutodoc... | Runs/002697_ProtChe... | 0 | ZINC000203686879 | 1 | 9 | 485 | Autodock4 | | 738000,000... | |
| 4 | ☑ | 4061 | | | Runs/002697_ProtChemRDKitP... | Runs/000485_ProtChemAutodoc... | Runs/002697_ProtChe... | 0 | ZINC000203686879 | 1 | 1 | 485 | Autodock4 | | 729000,000... | |
| 5 | ☑ | 2559 | | | Runs/002697_ProtChemRDKitP... | Runs/000485_ProtChemAutodoc... | Runs/002697_ProtChe... | 0 | ZINC000150601177 | 1 | 9 | 485 | Autodock4 | | 675000,000... | |
| 6 | ☑ | 2555 | | | Runs/002697_ProtChemRDKitP... | Runs/000485_ProtChemAutodoc... | Runs/002697_ProtChe... | 0 | ZINC000150601177 | 1 | 5 | 485 | Autodock4 | | 661000,000... | |
| 7 | ☑ | 4063 | | | Runs/002697_ProtChemRDKitP... | Runs/000485_ProtChemAutodoc... | Runs/002697_ProtChe... | 0 | ZINC000203686879 | 1 | 3 | 485 | Autodock4 | | 635000,000... | |
| 8 | ☑ | 4067 | | | Runs/002697_ProtChemRDKitP... | Runs/000485_ProtChemAutodoc... | Runs/002697_ProtChe... | 0 | ZINC000203686879 | 1 | 7 | 485 | Autodock4 | | 614000,000... | |
| 9 | ☑ | 4066 | | | Runs/002697_ProtChemRDKitP... | Runs/000485_ProtChemAutodoc... | Runs/002697_ProtChe... | 0 | ZINC000203686879 | 1 | 6 | 485 | Autodock4 | | 603000,000... | |
| 10 | ☑ | 4070 | | | Runs/002697_ProtChemRDKitP... | Runs/000485_ProtChemAutodoc... | Runs/002697_ProtChe... | 0 | ZINC000203686879 | 1 | 10 | 485 | Autodock4 | | 599000,000... | |
| 11 | ☑ | 4064 | | | Runs/002697_ProtChemRDKitP... | Runs/000485_ProtChemAutodoc... | Runs/002697_ProtChe... | 0 | ZINC000203686879 | 1 | 4 | 485 | Autodock4 | | 567000,000... | |
| 12 | ☑ | 2551 | | | Runs/002697_ProtChemRDKitP... | Runs/000485_ProtChemAutodoc... | Runs/002697_ProtChe... | 0 | ZINC000150601177 | 1 | 1 | 485 | Autodock4 | | 566000,000... | |
| 13 | ☑ | 2557 | | | Runs/002697_ProtChemRDKitP... | Runs/000485_ProtChemAutodoc... | Runs/002697_ProtChe... | 0 | ZINC000150601177 | 1 | 7 | 485 | Autodock4 | | 560000,000... | |
| 14 | ☑ | 4065 | | | Runs/002697_ProtChemRDKitP... | Runs/000485_ProtChemAutodoc... | Runs/002697_ProtChe... | 0 | ZINC000203686879 | 1 | 5 | 485 | Autodock4 | | 521000,000... | |
| 15 | ☑ | 2553 | | | Runs/002697_ProtChemRDKitP... | Runs/000485_ProtChemAutodoc... | Runs/002697_ProtChe... | 0 | ZINC000150601177 | 1 | 3 | 485 | Autodock4 | | 507000,000... | |
| 16 | ☑ | 2552 | | | Runs/002697_ProtChemRDKitP... | Runs/000485_ProtChemAutodoc... | Runs/002697_ProtChe... | 0 | ZINC000150601177 | 1 | 2 | 485 | Autodock4 | | 500000,000... | |
| 17 | ☑ | 5597 | | | Runs/002697_ProtChemRDKitP... | Runs/000485_ProtChemAutodoc... | Runs/002697_ProtChe... | 0 | ZINC000053229445 | 4 | 7 | 485 | Autodock4 | | 474000,000... | |
| 18 | ☑ | 5596 | | | Runs/002697_ProtChemRDKitP... | Runs/000485_ProtChemAutodoc... | Runs/002697_ProtChe... | 0 | ZINC000053229445 | 4 | 6 | 485 | Autodock4 | | 473000,000... | |
| 19 | ☑ | 5594 | | | Runs/002697_ProtChemRDKitP... | Runs/000485_ProtChemAutodoc... | Runs/002697_ProtChe... | 0 | ZINC000053229445 | 4 | 4 | 485 | Autodock4 | | 458000,000... | |
| 20 | ☑ | 5592 | | | Runs/002697_ProtChemRDKitP... | Runs/000485_ProtChemAutodoc... | Runs/002697_ProtChe... | 0 | ZINC000053229445 | 4 | 2 | 485 | Autodock4 | | 457000,000... | |
| 21 | ☑ | 5595 | | | Runs/002697_ProtChemRDKitP... | Runs/000485_ProtChemAutodoc... | Runs/002697_ProtChe... | 0 | ZINC000053229445 | 4 | 5 | 485 | Autodock4 | | 457000,000... | |

**Figure Appendix 13. Example of docking protocol output table showing predicted binding energies and pose information.**

Figure Appendix 14 and Figure Appendix 15 illustrate two representative cases generated using the Chimera visualizer tool: Brigatinib docked into ROS1 using the AutoDock-GPU protocol and GSK-3145095 docked into RIPK1 using the LeDock protocol. These visualizations display the ligand embedded within its predicted binding pocket, allowing for detailed structural evaluation of ligand–receptor interactions and spatial complementarity.

**Figure Appendix 14. Brigatinib docked into ROS1 using AutoDock-GPU.**

**Figure Appendix 15. GSK-3145095 docked into RIPK1 using LeDock.**

And subsequently, the remaining histograms are shown, representing the filtered outputs of LeDock (Figure Appendix 16) and AutoDock-GPU (Figure Appendix 17) for both ROS1 and RIPK1 targets after removing all docking poses with positive binding energies.

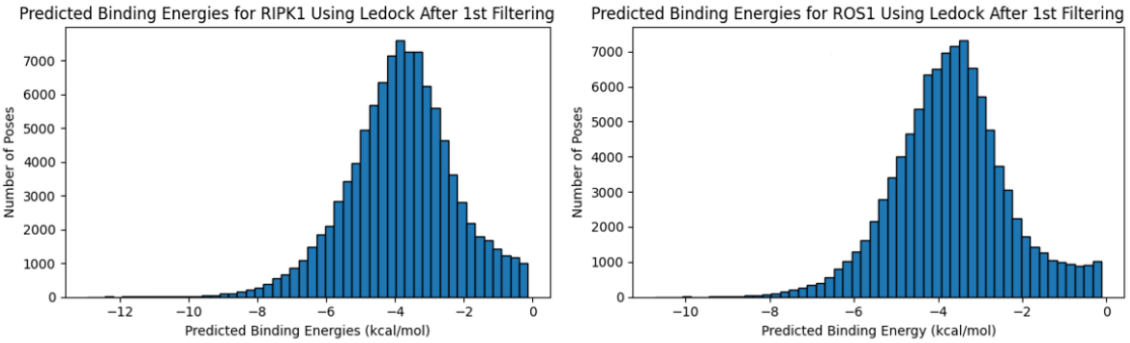



**Figure Appendix 16. Distribution of predicted binding energies for RIPK1 and ROS1 after first filtering step (LeDock).**

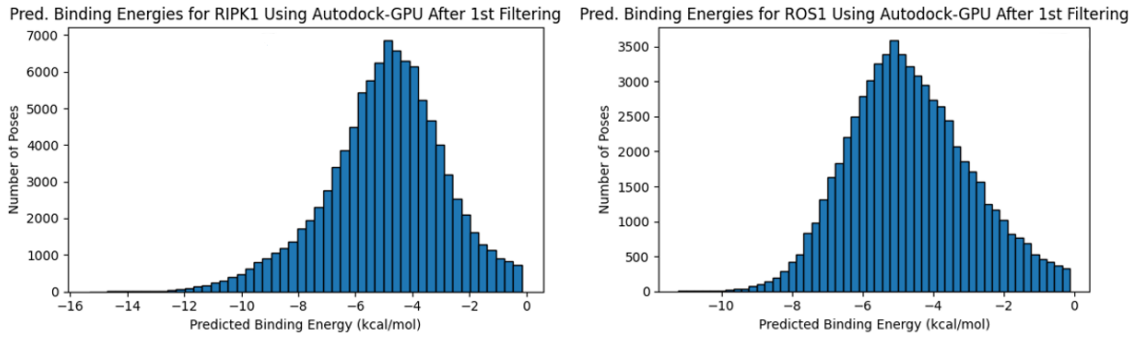



**Figure Appendix 17. Distribution of predicted (pred.) binding energies for RIPK1 and ROS1 after first filtering step (AutoDock-GPU).**

Additional scatter plots showing the correlation between predicted binding energies and ODDT Vina scores for LeDock and AutoDock-GPU and both targets are provided below from Figure Appendix 18 to Figure Appendix 21. These plots complement the main text by illustrating the full set of comparative scoring data not shown earlier due to space constraints.

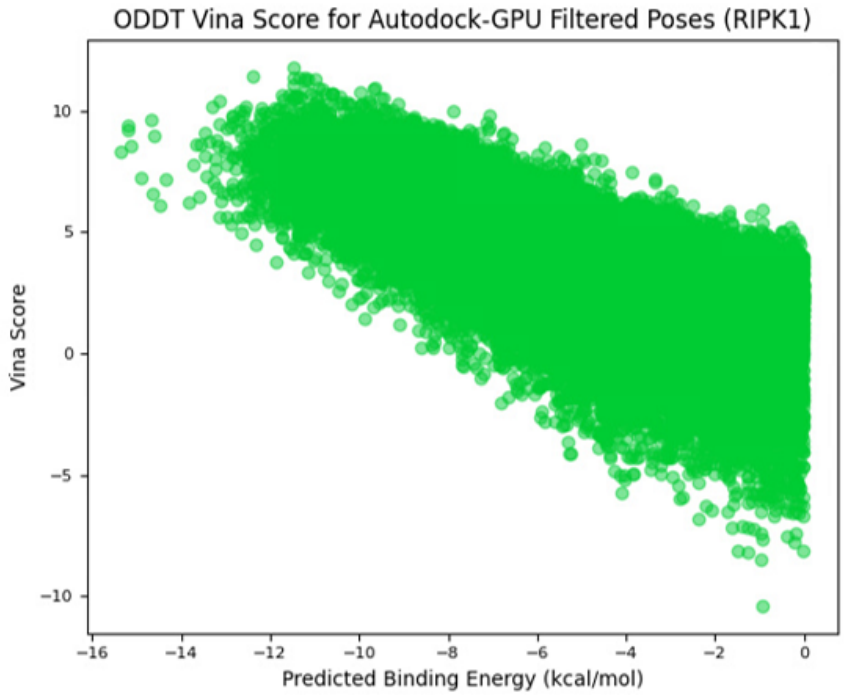



**Figure Appendix 18. Scatter plot showing the correlation between AutoDock-GPU binding energies and ODDT Vina scores for filtered RIPK1 ligand poses.**

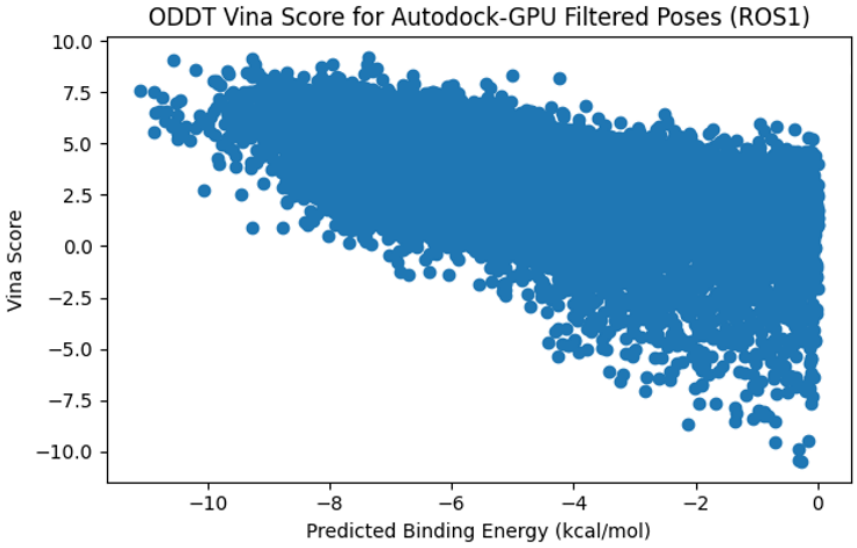



**Figure Appendix 19. Scatter plots showing the correlation between AutoDock-GPU binding energies and ODDT Vina scores for filtered ROS1 ligand poses.**

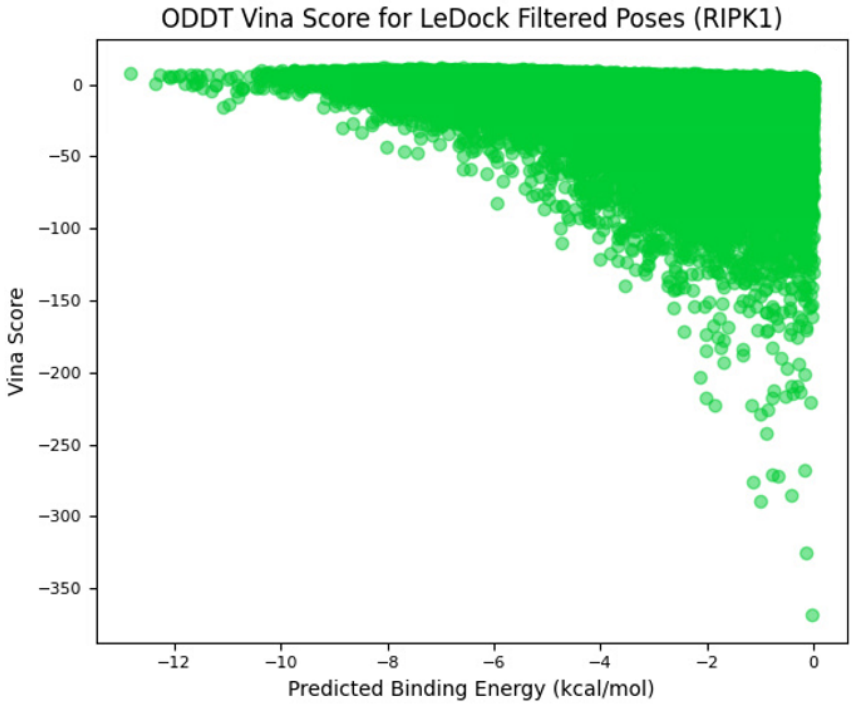



**Figure Appendix 20. Scatter plots showing the correlation between LeDock binding energies and ODDT Vina scores for filtered RIPK1 ligand poses.**

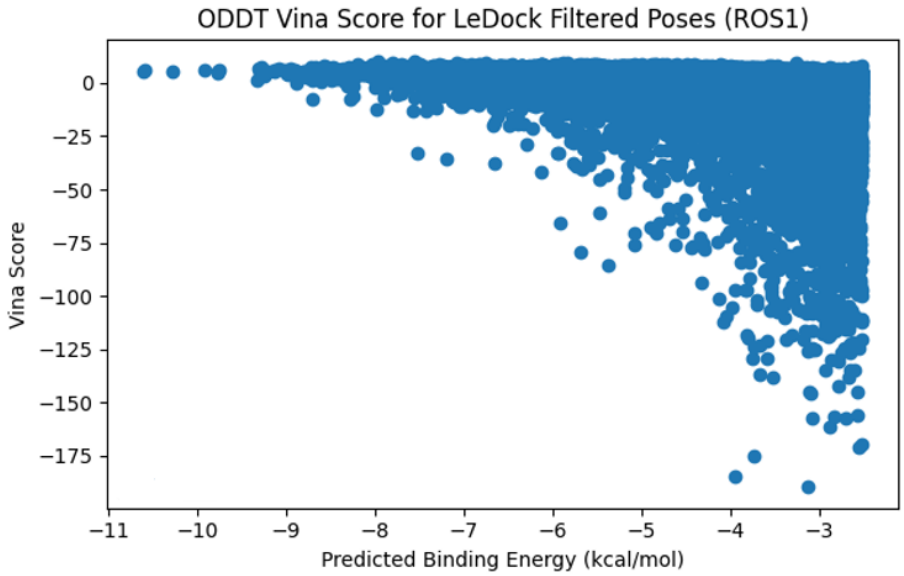



**Figure Appendix 21. Scatter plots showing the correlation between LeDock binding energies and ODDT Vina scores for filtered ROS1 ligand poses.**

The scatter plots from Figure Appendix 22 to Figure Appendix 25 display the relationship between the predicted binding energies from AutoDock-GPU and LeDock,

and ODDT Vina scores before and after applying the second filtering step, in which all poses with negative Vina scores were removed.

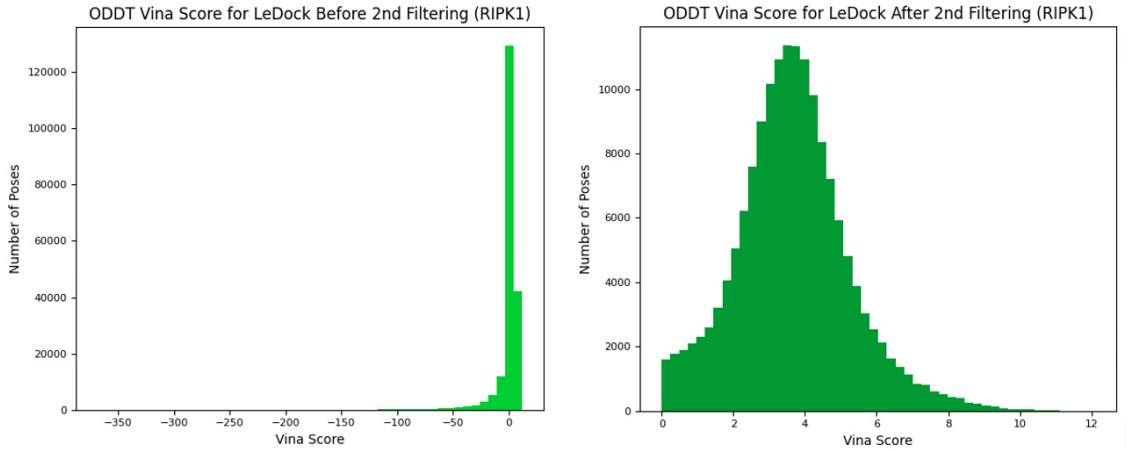



**Figure Appendix 22. Scatter plot showing the correlation between LeDock binding energies and ODDT Vina scores for filtered RIPK1 ligand poses after second filtering.**

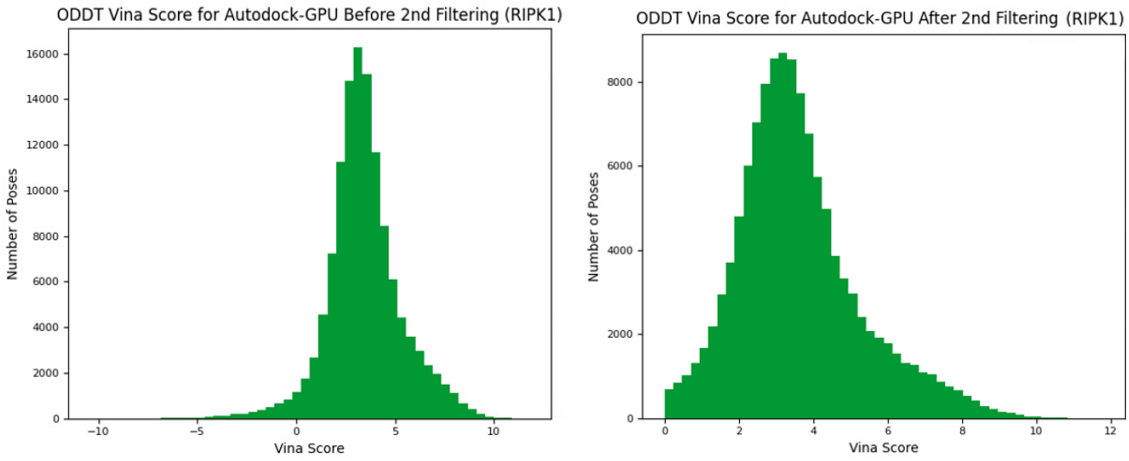



**Figure Appendix 23. Scatter plot showing the correlation between AutoDock-GPU binding energies and ODDT Vina scores for filtered RIPK1 ligand poses after second filtering.**

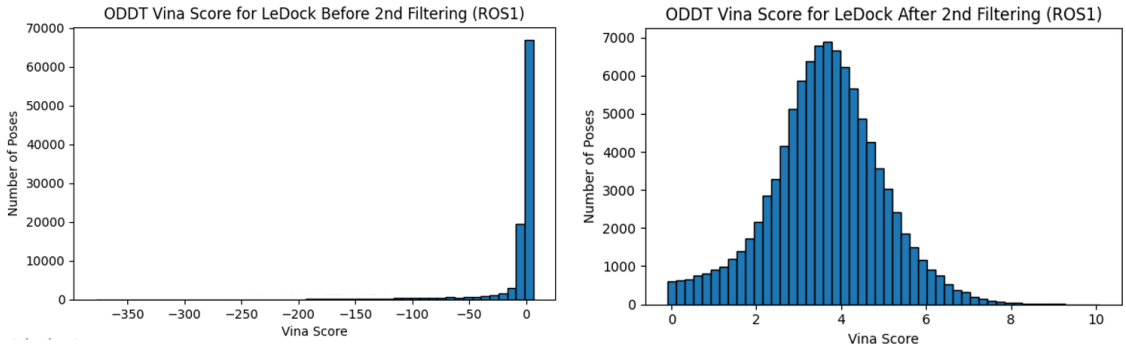



**Figure Appendix 24. Scatter plot showing the correlation between LeDock binding energies and ODDT Vina scores for filtered ROS1 ligand poses after second filtering.**

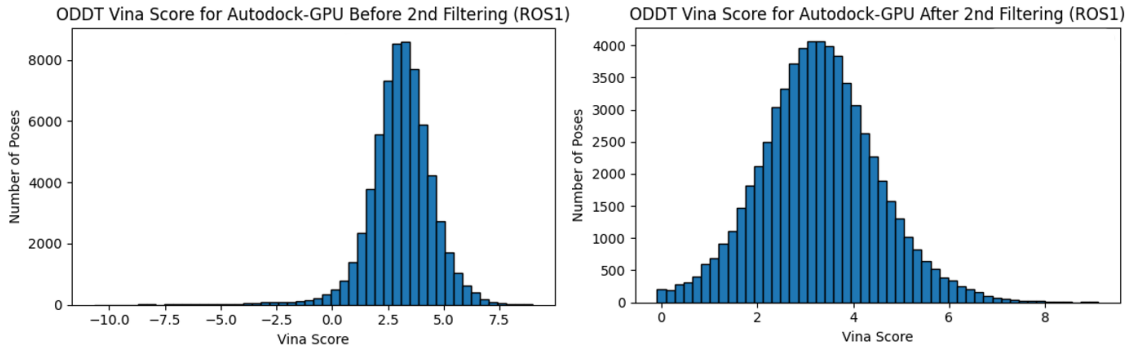



**Figure Appendix 25. Scatter plot showing the correlation between AutoDock-GPU binding energies and ODDT Vina scores for filtered ROS1 ligand poses after the second filtering.**

The results after the application of the third filtering step (binding energy ≤ –7 kcal/mol) are presented in Figure Appendix 26 and Figure Appendix 27.

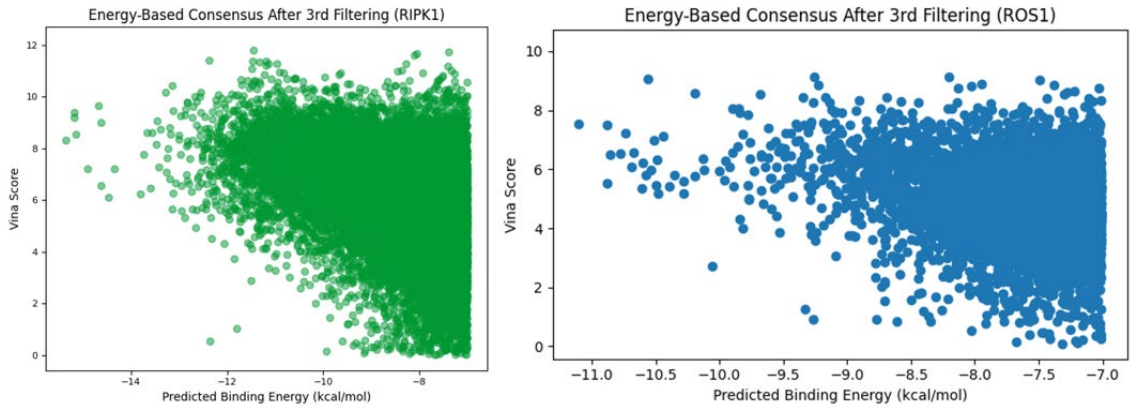



**Figure Appendix 26. Scatter plots of Vina scores versus predicted binding energies after the third filtering step based on the energy-based consensus configuration for RIPK1 (left) and ROS1 (right).**

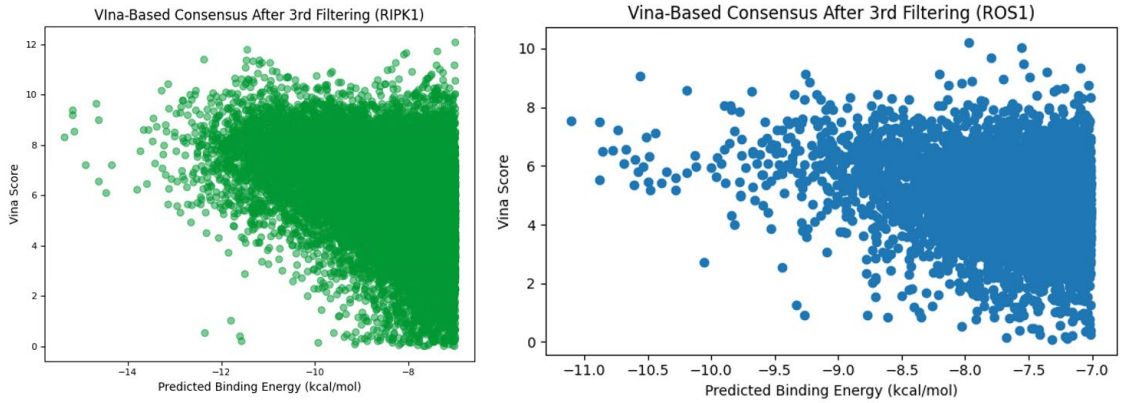



**Figure Appendix 27. Scatter plots of Vina scores versus predicted binding energies after the third filtering step based on the energy-based consensus configuration for RIPK1 (left) and ROS1 (right).**

For each combination of consensus and ranking configuration, Figure Appendix 28 through Figure Appendix 35 display the relationship between either the Vina score or the rank score and the predicted binding energy and Table Appendix 1 presents these same attributes for each of the reference ligands. This allows for a direct comparison of scoring behaviour and ranking outcomes across configurations.

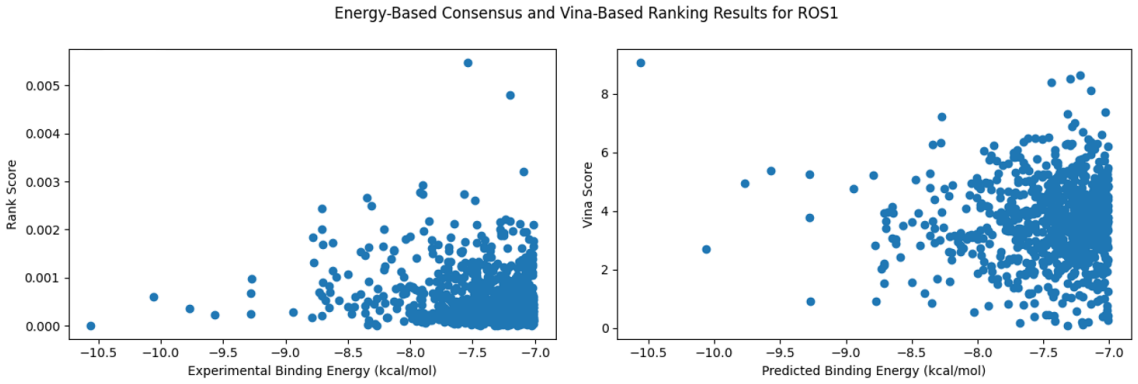



**Figure Appendix 28. Scoring and ranking metrics across energy-based consensus and Vina-based ranking for ROS1.**

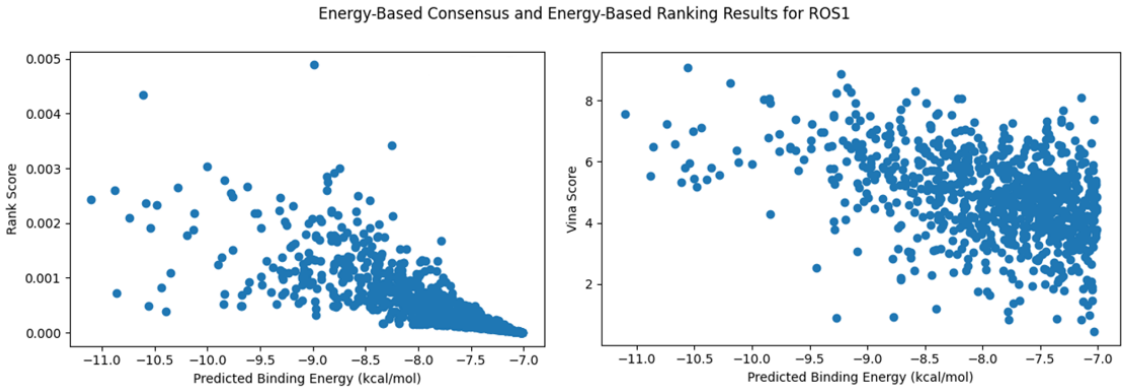



**Figure Appendix 29. Scoring and ranking metrics across energy-based consensus and energy-based ranking for ROS1.**

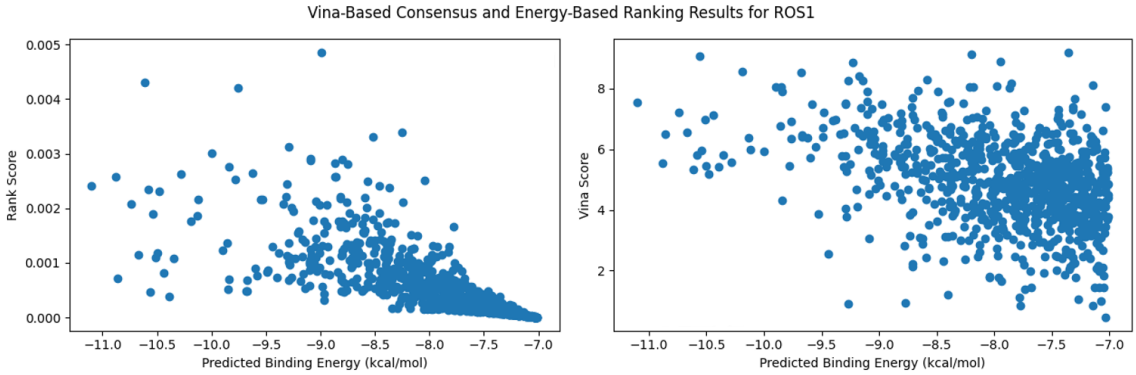



**Figure Appendix 30. Scoring and ranking metrics across Vina-based consensus and energy-based ranking for ROS1.**

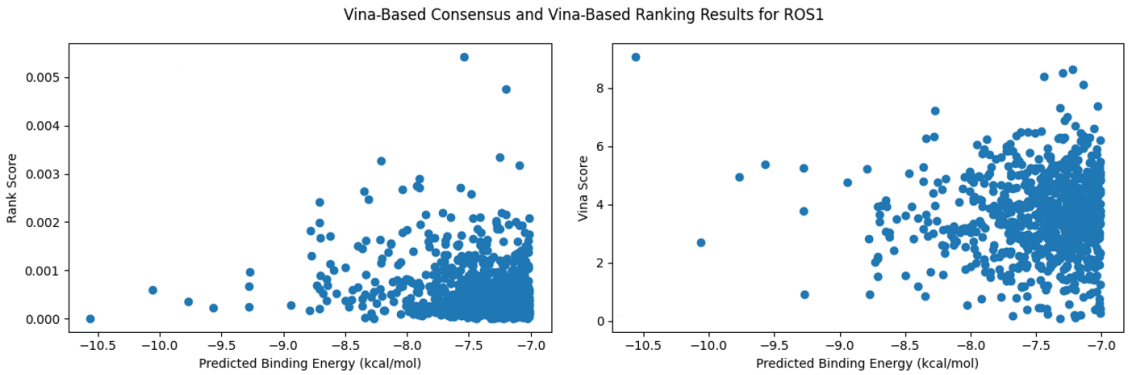



**Figure Appendix 31. Scoring and ranking metrics across Vina-based consensus and Vina-based ranking for ROS1.**

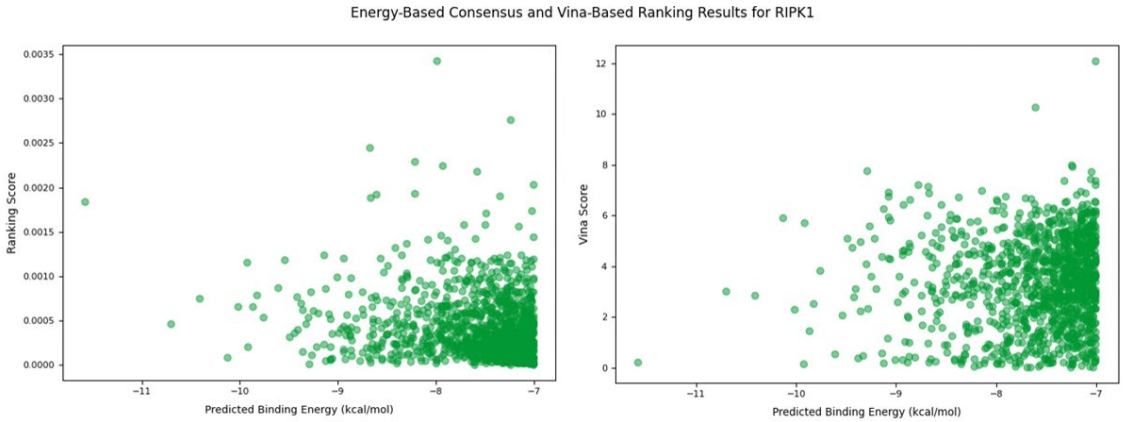



**Figure Appendix 32. Scoring and ranking metrics across energy-based consensus and Vina-based ranking for RIPK1.**

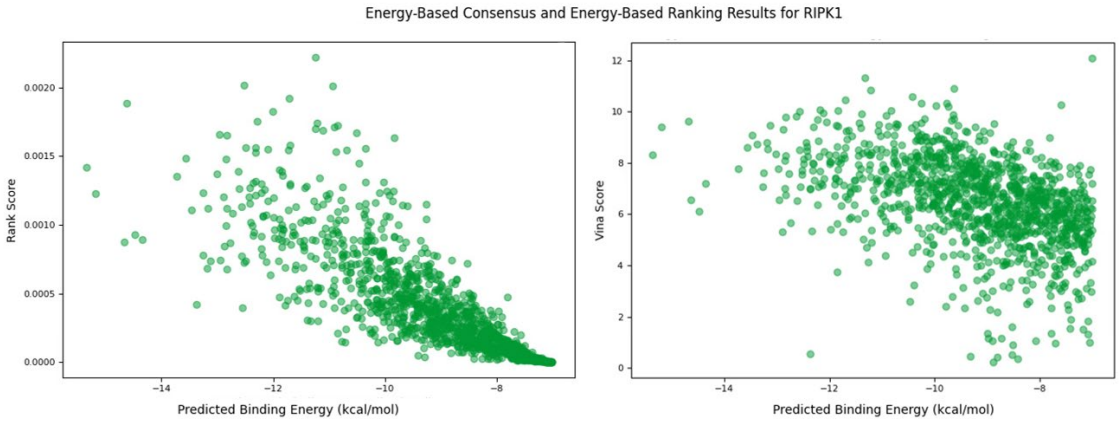



**Figure Appendix 33. Scoring and ranking metrics across energy-based consensus and energy-based ranking for RIPK1.**

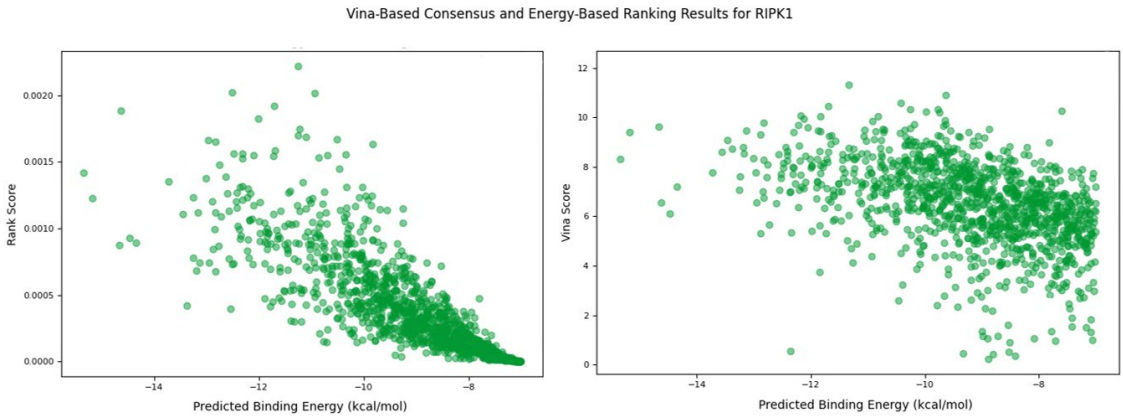



**Figure Appendix 34. Scoring and ranking metrics across Vina-based consensus and energy-based ranking for RIPK1.**

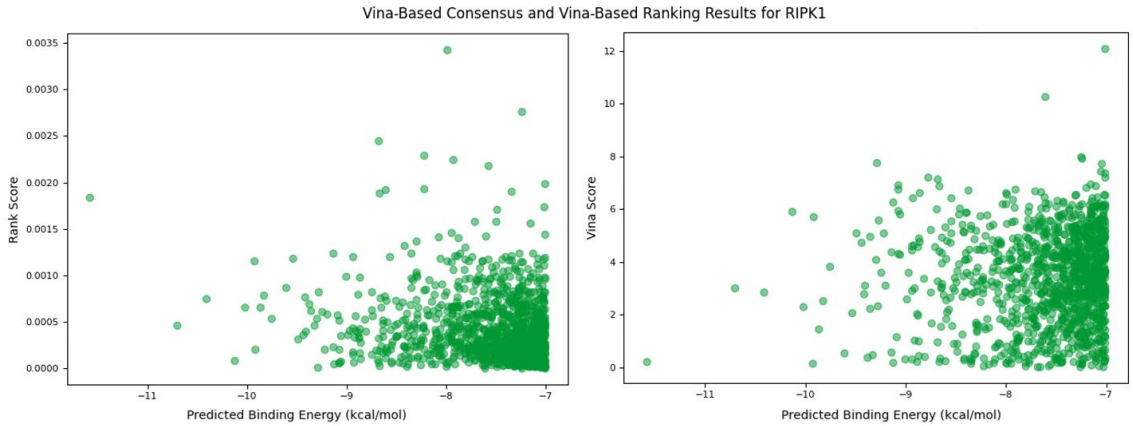



**Figure Appendix 35. Scoring and ranking metrics across Vina-based consensus and Vina-based ranking for RIPK1.**

| Protein Target | Drug Name /Formula | Energy Consensus + Energy Ranking | Energy Consensus + Vina Ranking | Vina Consensus + Energy Ranking | Vina Consensus + Vina Ranking |
|---|---|---|---|---|---|
| ROS1 | Entrectinib | Energy = -11,1 kcal/mol Vina Score = 7,39 Rank Score = 0,000773 | Energy = -7,42 kcal/mol Vina Score = 6,09 Rank Score = 0,00776 | Energy = -9,58 kcal/mol Vina Score = 7,49 Rank Score = 0,000766 | Energy = -7,42 kcal/mol Vina Score = 6,09 Rank Score = 0,00776 |
| | Brigatinib | Energy = -10,48 kcal/mol Vina Score = 7,16 Rank Score = 0,000867 | Energy = -7,27 kcal/mol Vina Score = 5,49 Rank Score = 0,00757 | Energy = -9,1 kcal/mol Vina Score = 7,16 Rank Score = 0,000861 | Energy = -7,27 kcal/mol Vina Score = 5,49 Rank Score = 0,00757 |
| | Ceritinib | Energy = -10 kcal/mol Vina Score = 5,94 Rank Score = 0,003033 | Energy = -7,43 kcal/mol Vina Score = 4,64 Rank Score = 0,00186 | Energy = -10,0 kcal/mol Vina Score = 5,94 Rank Score = 0,003008 | Energy = -7,43 kcal/mol Vina Score = 4,64 Rank Score = 0,00186 |
| | Staurosporin | Energy = -9,62 kcal/mol Vina Score = 4,16 Rank Score = 0,001372 | Energy = -7,61 kcal/mol Vina Score = 3,87 Rank Score = 0,000694 | Energy = -8,36 kcal/mol Vina Score = 4,16 Rank Score = 0,001362 | Energy = -7,61 kcal/mol Vina Score = 3,87 Rank Score = 0,000694 |
| | Crizotinib | Energy = -9,1 kcal/mol Vina Score = 6,09 Rank Score = 0,000086 | Energy = -7,41 kcal/mol Vina Score = 3,33 Rank Score = 0,000788 | Energy = -7,56 kcal/mol Vina Score = 5,10 Rank Score = 0,000242 | Energy = -7,41 kcal/mol Vina Score = 3,33 Rank Score = 0,000788 |

| | | | | | |
|---|---|---|---|---|---|
| | Nintedanib | Energy = -8,43 kcal/mol Vina Score = 5,10 Rank Score = 0,000244 | Energy = -7,27 kcal/mol Vina Score = 2,79 Rank Score = 0,0000697 | Energy = -11,1 kcal/mol Vina Score = 7,55 Rank Score = 0,002415 | Energy = -7,27 kcal/mol Vina Score = 2,79 Rank Score = 0,0000697 |
| | Foretinib | Energy = -7,76 kcal/mol Vina Score = 7,55 Rank Score = 0,002434 | Energy = -7,28 kcal/mol Vina Score = 1,58 Rank Score = 0,0000792 | Energy = -8,43 kcal/mol Vina Score = 6,61 Rank Score = 0,000827 | Energy = -7,28 kcal/mol Vina Score = 1,58 Rank Score = 0,0000792 |
| | Dasatinib | Energy = -7,34 kcal/mol Vina Score = 6,61 Rank Score = 0,000832 | Energy = -7,79 kcal/mol Vina Score = 1,06 Rank Score = 0,00000681 | Energy = -10,48 kcal/mol Vina Score = 5,19 Rank Score = 0,002313 | Energy = -7,79 kcal/mol Vina Score = 1,06 Rank Score = 0,00000681 |
| | Lorlatinib | Energy = -7,25 kcal/mol Vina Score = 5,19 Rank Score = 0,002331 | Energy = -8,5 kcal/mol Vina Score = 0,58 Rank Score = 0,00000636 | Energy = -8,69 kcal/mol Vina Score = 5,74 Rank Score = 0,001252 | Energy = -8,5 kcal/mol Vina Score = 0,58 Rank Score = 0,00000636 |
| | Repotrectinib | Energy = -7,11 kcal/mol Vina Score = 5,74 Rank Score = 0,001261 | Energy = -7,74 kcal/mol Vina Score = 0,43 Rank Score = 0,00000577 | | |
| **RIPK1** | $C_{20}H_{17}F_2N_5O_2$ | Energy = -8,7 kcal/mol Vina Score =7,8175 Rank Score =0,00018 | Energy = -7,63 kcal/mol Vina Score=12,0868 Rank Score =0,00031 | Energy = -8,38 kcal/mol Vina Score =7,8175 Rank Score =0,00011 | Energy = -7,63kcal/mol Vina Score =11,65 Rank Score =0,0003 |
| | $C_{23}H_{22}N_4O_6$ | Energy = -8,38 kcal/mol Vina Score =3,1974 Rank Score =0,0003 | Energy = -8,13 kcal/mol Vina Score =5,3694 Rank Score =0,0002 | Energy = -8,7 kcal/mol Vina Score =3,1974 Rank Score =0,00029 | Energy = -8,13kcal/mol Vina Score =4,34 Rank Score =0,0002 |
| | $C_{13}H_{12}ClN_3O_2$ | Energy = -7,73 kcal/mol Vina Score = 5,4043 Rank Score =0,00004 | Energy = -7,39kcal/mol Vina Score =4,7074 Rank Score =0,00019 | Energy = -7,73 kcal/mol Vina Score =5,54043 Rank Score =0,00008 | Energy = -7,39 kcal/mol Vina Score =4,70 Rank Score =0,00014 |
| | $C_{21}H_{19}N_3O_3S$ | Energy = -7,54 kcal/mol Vina Score = 3,9752 Rank Score =0,0001 | Energy = -7,5 kcal/mol Vina Score = 3,4468 Rank Score =0,00014 | Energy = -7,54kcal/mol Vina Score =3,9752 Rank Score =0,0012 | Energy = -7,5 kcal/mol Vina Score =3,4468 Rank Score =0,000144 |
| | $C_{21}H_{19}N_3O_4$ | Energy = -7,16 kcal/mol Vina Score =4,7074 Rank Score =0,00007 | Energy = -7,16 kcal/mol Vina Score =3,0718 Rank Score =0,0001 | Energy = -7,16 kcal/mol Vina Score =3,1974 Rank Score =0,001 | Energy = -7,16 kcal/mol Vina Score =3,0718 Rank Score =0,0001 |

| | | | | | |
|---|---|---|---|---|---|
| | $C_{24}H_{20}ClN_5O_3$ | Energy = -7,01 kcal/mol Vina Score =12,088 Rank Score =0,000067 | Energy = -7,01 kcal/mol Vina Score =3,1 Rank Score =0,00004 | Energy = -7,01 kcal/mol Vina Score =12,0868 Rank Score =0,003 | Energy = -7,01 kcal/mol Vina Score =3,01 Rank Score =0, 00004 |

**Table Appendix 1. Predicted binding energy, Vina score and rank score for reference ligands across all combinations of consensus and ranking strategies for ROS1 and RIPK1.**

Reports generated by the PLAPT viewer interface are also included in Figure Appendix 36 and Figure Appendix 37 to enhance clarity and completeness.

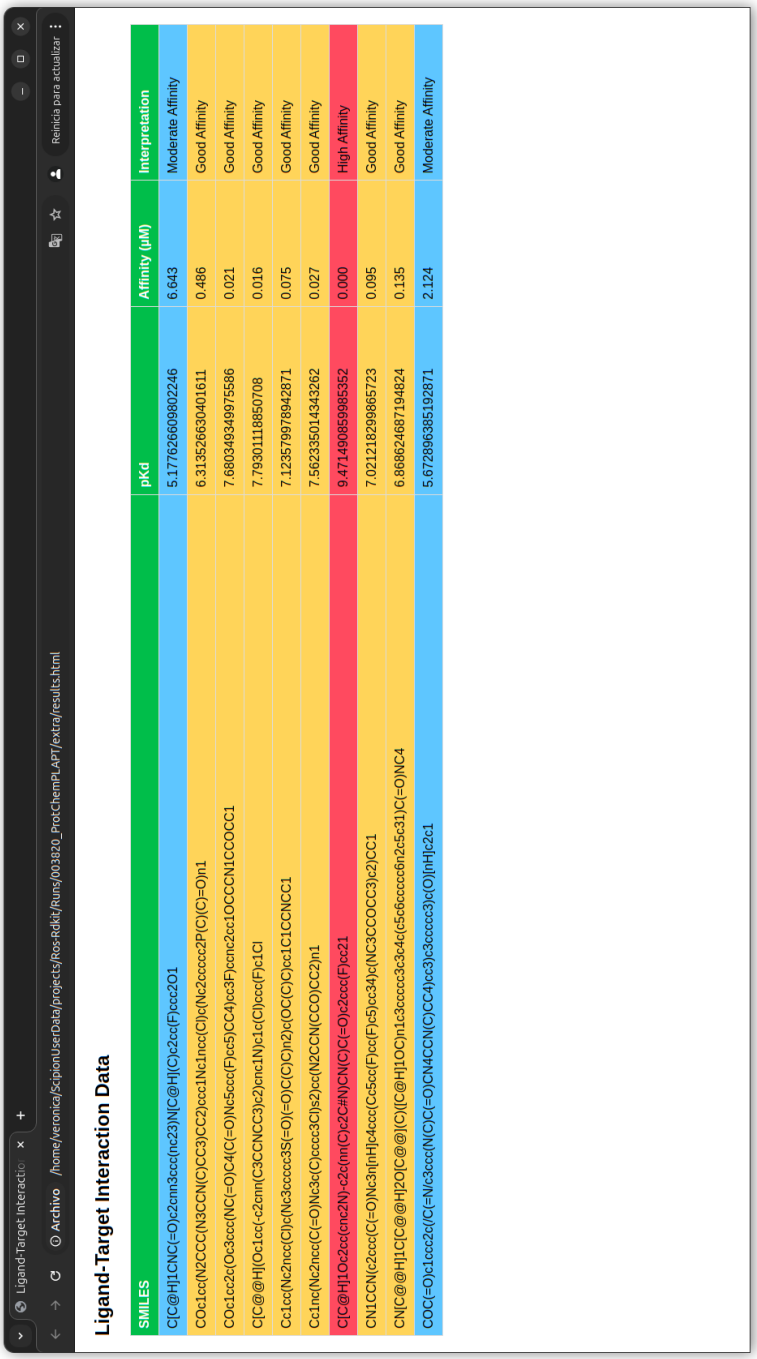
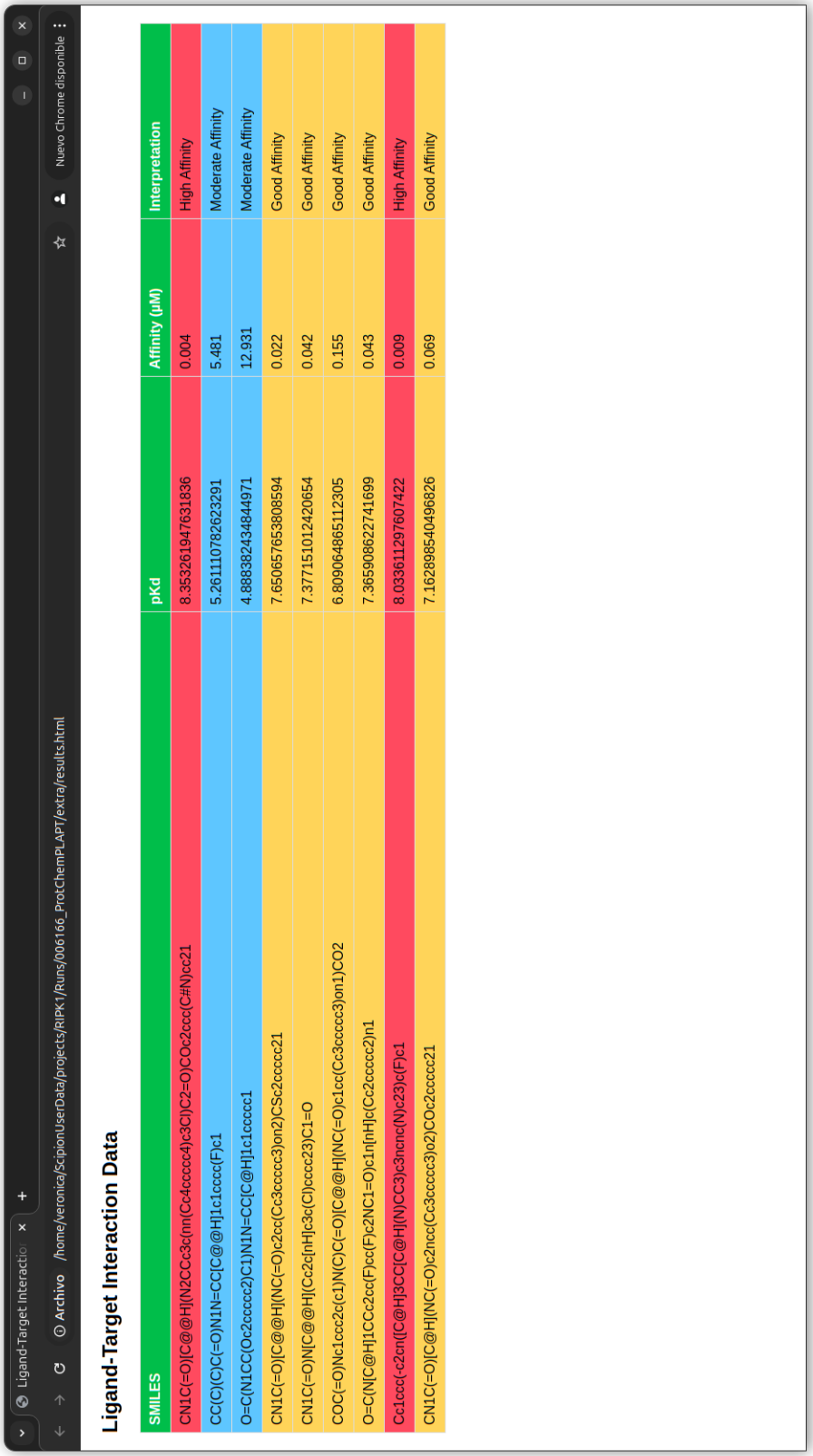



**Figure Appendix 36. Output table from the PLAPT viewer interface showing predicted affinity values and interaction classifications for ligand–ROS1 pairs.**




| SMILES | pKd | Affinity (µM) | Interpretation |
|---|---|---|---|
| CN1C(=O)[C@@H](N2CCc3c(nn(Cc4ccccc4)c3Cl)C2=O)COc2ccc(C#N)cc21 | 8.353261947631836 | 0.004 | High Affinity |
| CC(C)(C)C(=O)N1N=CC[C@@H]1c1ccccc(F)c1 | 5.261110782623291 | 5.481 | Moderate Affinity |
| O=C(N1CC(Oc2ccccc2)C1)N1N=CC[C@H]1c1ccccc1 | 4.888382434844971 | 12.931 | Moderate Affinity |
| CN1C(=O)[C@@H](NC(=O)c2cc(Cc3ccccc3)on2)CSc2ccccc21 | 7.650657653808594 | 0.022 | Good Affinity |
| CN1C(=O)N[C@@H](Cc2c[nH]c3c(Cl)ccccc23)C1=O | 7.377151012420654 | 0.042 | Good Affinity |
| COC(=O)Nc1ccc2c(c1)N(C)C(=O)[C@@H](NC(=O)c1cc(Cc3ccccc3)on1)CO2 | 6.809064865112305 | 0.155 | Good Affinity |
| O=C(N[C@H]1CCc2cc(F)cc(F)c2NC1=O)c1n[nH]c(Cc2ccccc2)n1 | 7.365908622741699 | 0.043 | Good Affinity |
| Cc1ccc(-c2cn([C@H]3CC[C@H](N)CC3)c3ncnc(N)c23)c(F)c1 | 8.033611297607422 | 0.009 | High Affinity |
| CN1C(=O)[C@H](NC(=O)c2ncc(Cc3ccccc3)o2)COc2ccccc21 | 7.162898540496826 | 0.069 | Good Affinity |

**Figure Appendix 37. Output table from the PLAPT viewer interface showing predicted affinity values and interaction classifications for ligand–RIPK1 pairs.**

Finally, Table Appendix 2 lists the ranking positions of the reference ligands among all drugs, based on predicted affinity values from the Pafnucy and PLAPT models. Rankings are sorted in descending order of predicted affinity, with lower rank numbers indicating stronger predicted binding.

| Protein Target | Drug Name/Formula | Pafnucy Position | PLAPT Position |
|---|---|---|---|
| ROS1 | Entrectinib | 1002 | 678 |
| | Brigatinib | 68 | 604 |
| | Ceritinib | 124 | 299 |
| | Staurosporin | 154 | 398 |
| | Crizotinib | 255 | 100 |
| | Nintedanib | 631 | 901 |
| | Foretinib | 296 | 136 |
| | Dasatinib | 478 | 169 |
| | Lorlatinib | 550 | 2 |
| | Repotrectinib | 878 | 1158 |
| RIPK1 | $C_{20}H_{17}F_2N_5O_2$ | 171 | 304 |
| | $C_{23}H_{22}N_4O_6$ | 191 | 397 |
| | $C_{13}H_{12}ClN_3O_2$ | 671 | 214 |
| | $C_{21}H_{19}N_3O_3S$ | 230 | 189 |
| | $C_{21}H_{19}N_3O_4$ | 302 | 346 |
| | $C_{24}H_{20}ClN_5O_3$ | 74 | 57 |
| | $C_{19}H_{19}N_3O_2$ | 391 | 526 |
| | $C_{30}H_{28}F_2N_6O_4$ | 544 | 124 |
| | $C_{14}H_{17}FN_2O$ | 604 | 422 |

**Table Appendix 2. Ranking positions of reference ligands according to predicted affinity by Pafnucy and PLAPT.**