PLOS COMPUTATIONAL BIOLOGY

EDUCATION

Using interactive Jupyter Notebooks and BioConda for FAIR and reproducible biomolecular simulation workflows

Genís Bayarri[1], Pau Andrio[2], Josep Lluís Gelpí[2,3], Adam Hospital[1]*, Modesto Orozco[1,3]*

1 Institute for Research in Biomedicine (IRB Barcelona), the Barcelona Institute of Science and Technology, Barcelona, Spain, **2** Barcelona Supercomputing Center (BSC), Barcelona, Spain, **3** Department of Biochemistry and Biomedicine, University of Barcelona, Barcelona, Spain

* adam.hospital@irbbarcelona.org (AH); modesto.orozco@irbbarcelona.org (MO)

Citation: Bayarri G, Andrio P, Gelpí JL, Hospital A, Orozco M (2024) Using interactive Jupyter Notebooks and BioConda for FAIR and reproducible biomolecular simulation workflows. PLoS Comput Biol 20(6): e1012173. https://doi.org/10.1371/journal.pcbi.1012173

Editor: B. F. Francis Ouellette, bioinformatics.ca, CANADA

Published: June 20, 2024

Funding: Authors acknowledge funding from the BioExcel Centre of Excellence for Computational Biomolecular Research (BioExcel-2 [823830]; BioExcel-3 [European Union: 101093290; Ministerio de Ciencia e Innovación: PCI2022-134976-2]) the Spanish Ministry of Science [RTI2018-096704-B-100, PID2021-122478NB-I00] the Instituto de Salud Carlos III–Instituto Nacional de Bioinformatica, Fondo Europeo de Desarrollo Regional [ISCIII PT 17/0009/0007], the European Regional Development Fund, ERFD Operative Programme for Catalunya, the Catalan Government AGAUR [SGR2021 00863] and the MDDB: Molecular Dynamics Data Bank European Repository for Biosimulation Data [101094651], all

Abstract

Interactive Jupyter Notebooks in combination with Conda environments can be used to generate FAIR (Findable, Accessible, Interoperable and Reusable/Reproducible) biomolecular simulation workflows. The interactive programming code accompanied by documentation and the possibility to inspect intermediate results with versatile graphical charts and data visualization is very helpful, especially in iterative processes, where parameters might be adjusted to a particular system of interest. This work presents a collection of FAIR notebooks covering various areas of the biomolecular simulation field, such as molecular dynamics (MD), protein–ligand docking, molecular checking/modeling, molecular interactions, and free energy perturbations. Workflows can be launched with myBinder or easily installed in a local system. The collection of notebooks aims to provide a compilation of demonstration workflows, and it is continuously updated and expanded with examples using new methodologies and tools.

Introduction

Scientific research is inexorably moving to an open and reproducible format [1]. Increased research transparency and reproducibility have a direct effect on the credibility of the scientific literature published. Reproducibility in computational science has greatly improved in the last few years [2,3], mainly boosted by the data science community [4]. Life sciences and computational biology research have recently started to adopt these methodologies [5–8]. Furthermore, a new movement intending to adapt the Findable, Accessible, Interoperable and Reusable/Reproducible (FAIR) data principles [9] to research software development has emerged [10–13]. Using standards and state-of-the-art tools, this software FAIRification ensures not only reproducibility, but also *Findability* (how to find), *Accessibility* (how to access), *Interoperability* (how to integrate), and *Reusability* (how to use).

Regrettably, the field of biomolecular simulation is known for its lack of standards and best practices, thus hindering the generation of open, interoperable, and reproducible science. Several recent initiatives have attempted to start community-driven processes to define best

of them awarded to M.O. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Competing interests: The authors have declared that no competing interests exist.

practices for software and workflow development and data sharing in the field [6,14–16]. The BioExcel Centre of Excellence for Computational Biomolecular Research (BioExcel CoE, https://bioexcel.eu/) is leading one such initiative. BioExcel Building Blocks (BioBB) [17] is a new computational library with particular attention to interoperability, accessibility, and reproducibility/reusability in biomolecular simulation workflows. Designed following the recommendations of the ELIXIR European Life Science infrastructure and the FAIR principles for software development, the library provides a set of interoperable units based on a collection of Python wrappers encapsulating software components. It is freely accessible from GitHub and BioConda packages [18], installable via Conda, Docker and Singularity containers, and compatible with several workflow languages [19].

Jupyter Notebooks are documents shared in an open-source web application (computational notebook), and they are compatible with live code, equations, visualizations, and text. They are now routinely used for documentation purposes in code development processes [20], openness/reproducibility [21,22], and for hands-on teaching [23–26]. During the COVID-19 pandemic (2020), they emerged as fantastic tools for online training events, with the possibility to use them as tutorials comprising interactive programming code accompanied by text information and/or documentation, versatile graphical charts, and data visualization. Furthermore, the combination of Jupyter Notebooks with easy installation of software dependencies using packaging systems such as Conda [27] makes these workflows highly shareable and reproducible.

Here, we present a collection of reproducible, FAIR, easy-to-install, and highly documented Jupyter Notebooks implementing common biomolecular simulations workflow tutorials, developed with the combination of the BioBB library and BioConda packages, ready to be used for training purposes.

Results/Evaluation

The collection of Jupyter Notebooks can be accessed from the workflows section of the main BioBB website: https://mmb.irbbarcelona.org/biobb/workflows. Each tutorial has an independent block with a short description, version, WorkflowHub entries, available deployment sites, source code, and documentation (Fig 1).

The current version (2024.1) provides a collection of 17 workflows covering various areas of the biomolecular simulation field, including molecular dynamics (MD), protein–ligand docking, molecular checking/modeling, molecular interactions, and free energy perturbations. An exhaustive list of available workflows and where they can be found is given in Table A in S1 Data.

All the notebooks share a uniform header, with a title and description, a list of the BioBB modules and auxiliary libraries used, the command lines required to install and launch the workflow, and a clickable index of the pipeline steps (Fig A in S1 Data). Following this header, the pipelines encoded in the notebooks are fragmented and explained step by step. Each step process is enclosed in a Jupyter cell, preceded by documentation about what the step is doing, and followed by a graphical inspection of intermediate results whenever possible (Fig 2). This format allows the practice of reading, understanding, and executing the pipeline gradually, moving from top to bottom.

In addition, the use of the BioBB library makes the workflows easier to understand, as all execution step cells follow the same uniform syntax (Fig 2 inline). Each of these cells imports a specific building block, defines the inputs/outputs (as regular disk files) and properties (method variables) of the tool wrapped, and finally launches the execution.

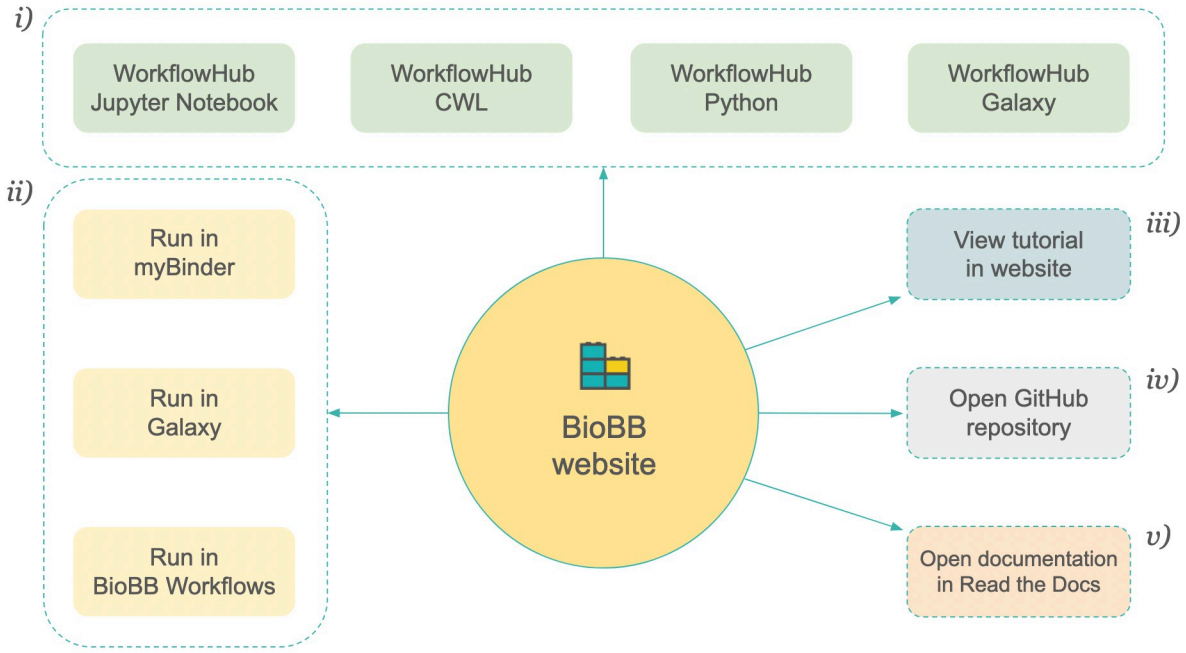

Fig 1. Information available on the BioBB website for the workflow tutorials, including links to: (i) workflow versions registered and available in WorkflowHub (Jupyter Notebook, CWL, Python and Galaxy); (ii) ways to directly launch the workflow (Jupyter Notebook myBinder, Galaxy and BioBB Workflows [28] website); (iii) tutorial in web version (HTML); (iv) source code in GitHub; and (v) documentation in Read the Docs.

https://doi.org/10.1371/journal.pcbi.1012173.g001

To demonstrate the utility of our interactive Jupyter Notebooks, 4 distinct examples, briefly described in the following sections, were selected.

Example 1: Protein MD setup (GROMACS)

Target audience: Users interested in MD simulations with little or even no knowledge about the method.

Examples of use: Prepare (setup) and run your first MD simulation for a small globular protein structure using the GROMACS MD package.

GitHub URL: https://github.com/bioexcel/biobb_wf_md_setup

The first example tutorial [29] aims to illustrate the process of setting up an MD simulation system containing a protein with GROMACS MD engine [30], using the Lysozyme protein (PDB code 1AKI) as input. The workflow is strongly based on the pipeline included in the collection of MD tutorials written by Justin Lemkul and available on-line here: https://www.mdtutorials.com/gmx/lysozyme/index.html [31].

The workflow is divided into 14 sections, including retrieving the PDB structure from the PDB database [32], preparing the structure (Fig 3A), developing an MD topology and system box with explicit water molecules and counterions, energetically minimizing and equilibrating the system (Fig 3B), running a short, unrestrained MD simulation, and finally post-processing (Fig 3C) and visualizing the short trajectory generated (Fig 3D).

Inspection of intermediate results (Fig 3) is a useful functionality in Jupyter Notebooks. As an example in this particular case, visualization of the outputs for the energy minimization or system equilibration process is a direct indicator of convergence problems in the system energy and/or box, which could suggest that these processes should be extended to allow the

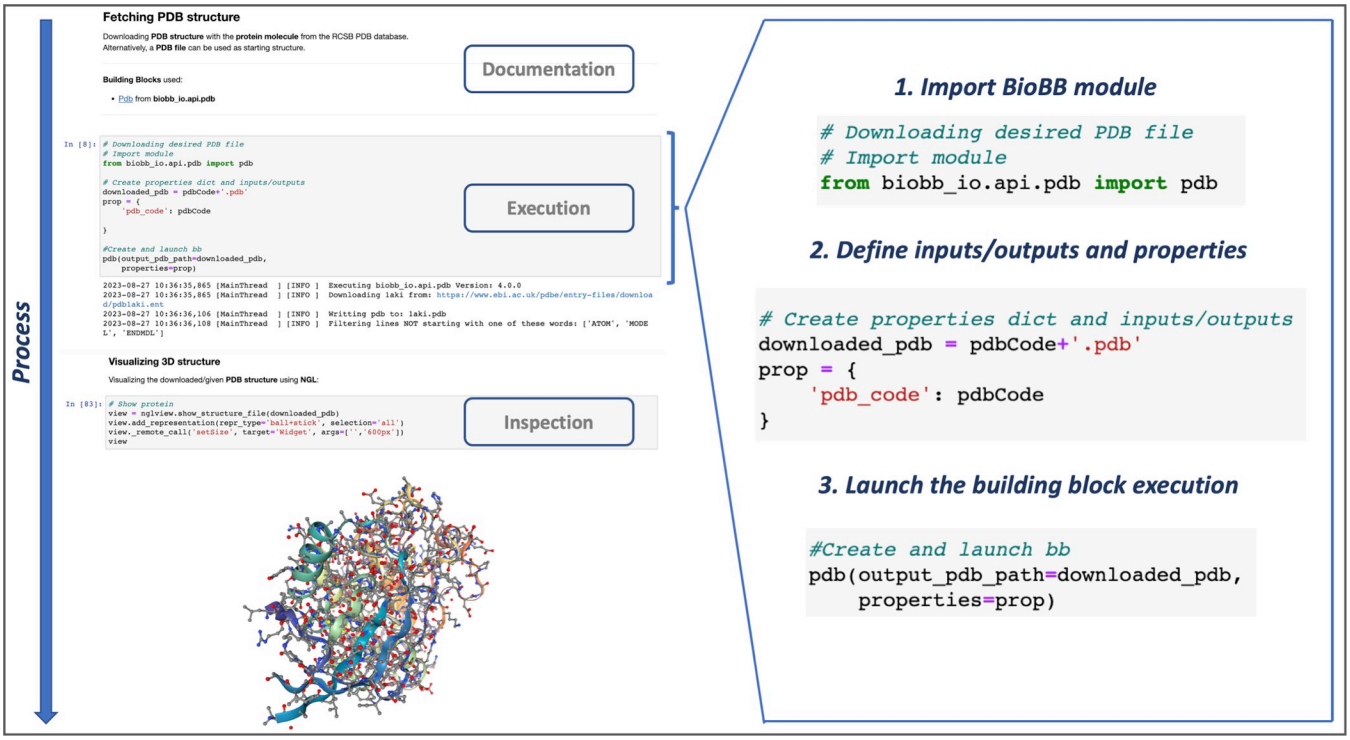

Fig 2. Uniformity in the Jupyter Notebooks collection: in the pipeline process, with markdown cell for documentation followed by the execution cell and the graphical inspection of the intermediate results (left); and in the cell execution, using the BioBB syntax of importing the module, defining inputs/outputs and properties, and launching the execution for all processes run in the workflow (inline, right).

https://doi.org/10.1371/journal.pcbi.1012173.g002

system to relax. The final root mean square deviations (RMSd) can also denote that the simulation needs more time to converge.

Associated with the convergence, it is noteworthy that the time scales used in this demonstration workflow are shorter than those typically used in the field, for the sake of time in running the whole workflow. For example, a common equilibration process can easily reach tens of nanoseconds, whereas a common production MD can go up to the millisecond time scale. Although these variables can be easily adjusted in the workflow, it is important to understand that these notebooks are designed to be used as demonstrators and not as production scripts.

Example 2: Protein–ligand docking

Target audience: Users interested in protein–ligand docking processes with little or even no knowledge about the method.

Examples of use: Find possible binding sites on the surface of a protein structure, explore the suitability for a particular ligand to dock at one of the discovered regions, and obtain a final predicted protein–ligand complex structure.

GitHub URL (Protein–ligand docking workflows):
https://github.com/bioexcel/biobb_wf_virtual-screening

GitHub URL (Protein–ligand docking workflow, *fpocket* version):
https://github.com/bioexcel/biobb_wf_virtual-screening/tree/master/biobb_wf_virtual-screening/notebooks/fpocket

The second example [33] illustrates the process of protein–ligand docking, a prediction of the position and orientation of a small molecule when bound to a protein receptor or enzyme.

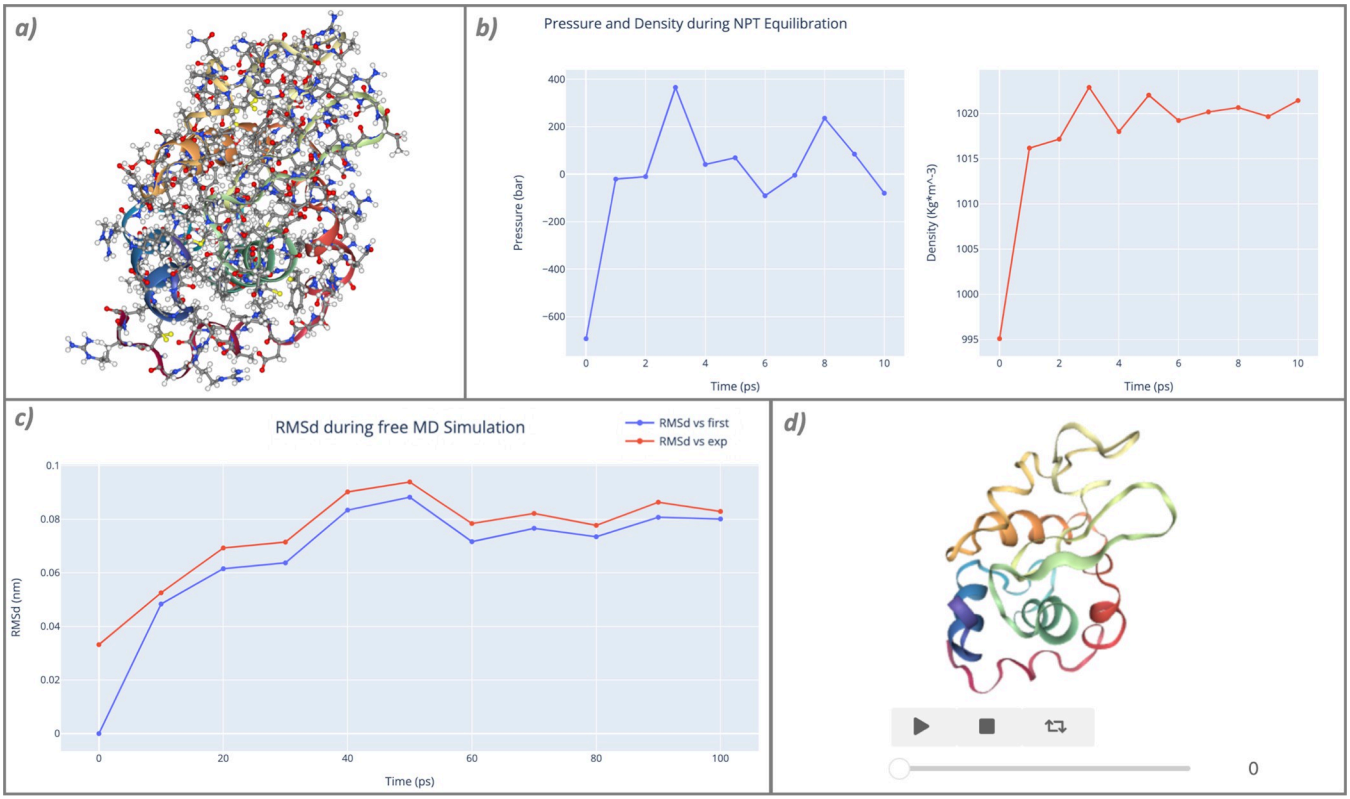

Fig 3. Intermediate results extracted from the "GROMACS Protein MD Setup" workflow. (a) PDB structure with missing atoms added, ready to be used as MD input; (b) pressure and density box parameters measured over time in the final NPT equilibration process (10ps); (c) RMSd of the snapshots included in the final trajectory against the first snapshot (blue) and the original experimental structure (red); and (d) interactive visualization of the final trajectory using the simpletraj tool.

https://doi.org/10.1371/journal.pcbi.1012173.g003

The particular example used is the Mitogen-activated protein kinase 14 (p38-α) protein (PDB code 3HEC), a well-known protein kinase enzyme, in complex with the FDA-approved Imatinib (PDB Ligand code STI, DrugBank [34] Ligand Code DB00619), a small molecule kinase inhibitor used to treat certain types of cancer. The tutorial guides the user through the process of identifying the active site cavity (pocket) without previous knowledge using fpocket [35], and the final prediction of the protein–ligand complex using the AutoDock Vina program [36].

The workflow is divided into 17 sections, including retrieving the PDB structure from the PDB database, computing the protein cavities (Fig 4A), generating a box around a selected cavity (Fig 4B), downloading the ligand, preparing the 2 molecules (protein and ligand) for the docking process (adding atomistic charges and formatting the files), running the docking task, extracting a chosen docking pose, superposing the final ligand pose to the target protein structure, and comparing the final result with an existing experimental structure to validate the workflow (Fig 4C).

This workflow adds more interactivity than the previous one, including the possibility for the user to choose the pocket to include in the final docking run, and also the ligand pose to keep as a final result. These selections are implemented as dropdown lists in the Jupyter Notebook. However, this interactivity makes the workflow unsuitable to be used in an unsupervised way, as stated at the beginning of the notebook with a warning banner.

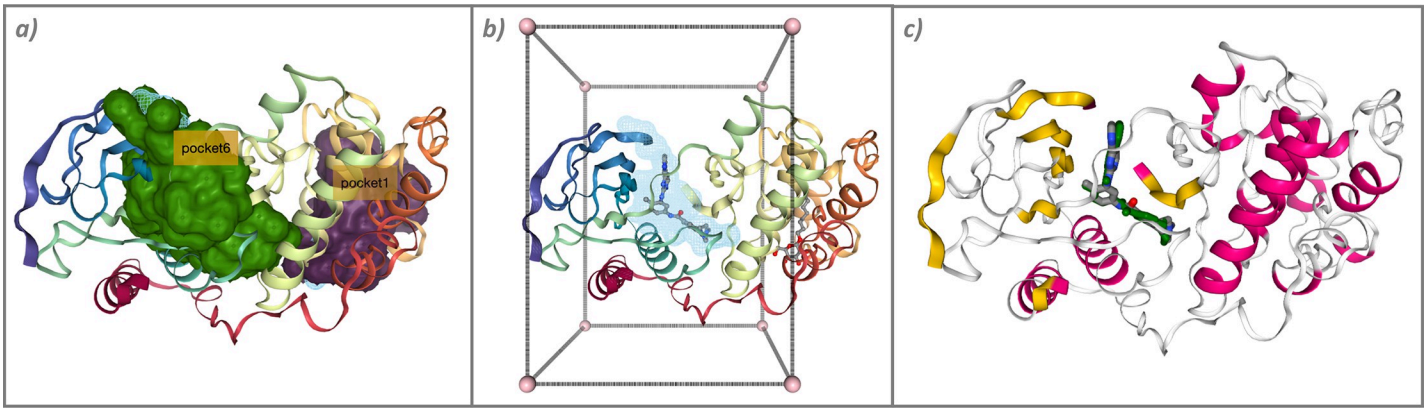

Fig 4. Intermediate results extracted from the "Protein–ligand Docking" workflow. (a) Pockets identified on the surface of the protein by the fpocket tool; (b) box including the protein pocket to be used in the docking process; and (c) final comparison of the chosen ligand pose against the experimental structure.

https://doi.org/10.1371/journal.pcbi.1012173.g004

Example 3: Protein coarse-grained flexibility

Target audience: Users interested in fast (coarse-grained) theoretical methods to explore protein dynamics and flexibility with little or even no knowledge about the methods.

Examples of use: Generate a set of predicted 3D conformations from a determined protein and explore the main flexible regions and global dynamics without spending too much time and computational resources.

GitHub URL: https://github.com/bioexcel/biobb_wf_flexserv

The third tutorial [37] aims to illustrate the process of generating protein conformational ensembles from 3D structures and analyzing their molecular flexibility. The notebook reproduces the workflow integrated into the FlexServ web-based tool for the analysis of protein flexibility [38]. The workflow incorporates powerful protocols for the coarse-grained (CG) determination of protein dynamics using different versions of Normal Mode Analysis (NMA), Brownian dynamics (BD), and Discrete Dynamics (DMD). It also integrates a set of flexibility analyses using a large variety of metrics, including basic geometrical analysis, B-Factors, essential dynamics, stiffness analysis, collectivity measures, Lindemann's indexes, residue correlation, chain-correlations, dynamic domain determination, and hinge point detections, all of them computed with the pcasuite tool [39,40]. The particular structure used is the Ribosomal Protein S15 from Bacillus Stearothermophilus (PDB code 1A32).

The workflow is divided into 2 main sections, namely CG generation of conformational ensembles, and flexibility analysis of the resulting pseudo-trajectories. The first part involves computing the ensembles using 3 CG methods: NMA, BD, and DMD while the second part uses the PCA statistical method to extract the essential dynamics [41] from the ensembles and obtain a collection of flexibility descriptors, which are graphically represented using NGL [42,43] and plotly [44] tools. The features obtained from these analyses facilitate the identification of the protein flexible domains through the components of the eigenvalue (Fig 5A), B-Factors, or hinge point prediction (Fig 5B). To graphically examine how far the pseudo-trajectories fall from covering the protein conformational landscape explored by MD, the workflow includes a final comparison of all the CG-generated ensembles with an atomistic MD simulation, (Fig 5C).

One particularity of this workflow is the rich markdown documentation included. Each CG method is complemented by an extended explanation, with equations embedded in mathematical notation (Fig 5D). Therefore, the notebook offers a reference to the CG methods basics,

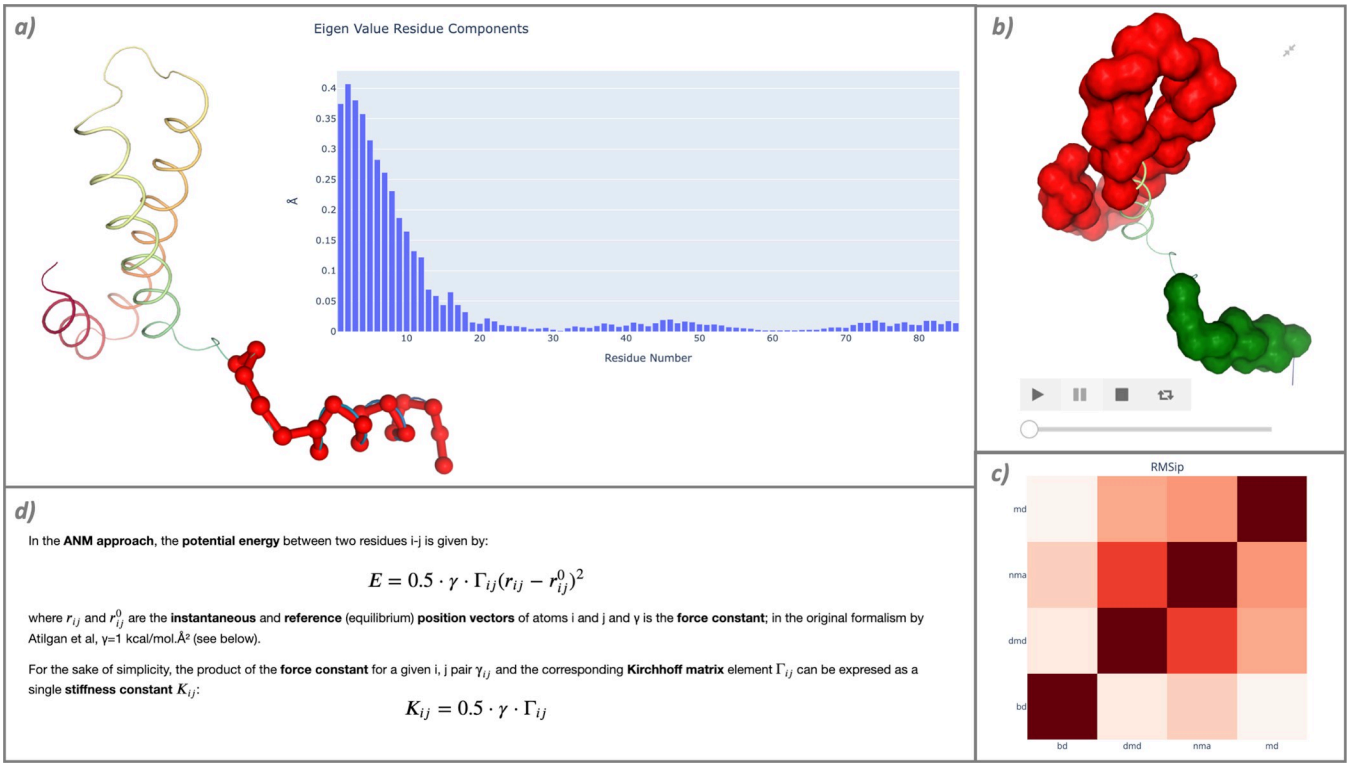

Fig 5. Markdown documentation and intermediate results extracted from the "Macromolecular Coarse-Grained Flexibility" workflow. (a) Eigenvalue residue components analysis, indicating the residues contributing the most to the key essential deformations of the protein. An associated NGL widget displays these residues in ball and stick representation (red); (b) domain decomposition analysis displayed along the pseudo-trajectory with the simpletraj tool; (c) RMS inner product (RMSip) for all the CG pseudo-trajectories against an atomistic MD simulation; and (d) extract of documentation included for the NMA CG method.

https://doi.org/10.1371/journal.pcbi.1012173.g005

while simultaneously giving the user the possibility to run and explore the generated ensembles.

Example 4: Molecular interaction potentials

Target audience: Users interested in exploring molecular interactions (protein–ligand, protein–protein) for a particular system. Basic knowledge about electrostatic interactions is recommended.

Examples of use: Place a first shell of water molecules in the energetically most favorable positions of the protein surface; explore the interaction properties of a protein surface regions through the affinity for negative, positive, or hydrophobic residues; identify the residue/atom which is contributing the most to a particular protein–ligand or protein–protein complex interaction, and evaluate the interaction energy.

GitHub URL: https://github.com/bioexcel/biobb_wf_cmip

The last tutorial used to exemplify the collection of workflows aims to illustrate the process of computing classical molecular interaction potentials from protein structures [45]. It includes molecular interaction potential (MIP) grids, protein–protein/ligand interaction potentials, and protein titration. The particular structures used are the Lysozyme protein (PDB code 1AKI), the Epidermal Growth Factor Receptor kinase domain (PDB code 4HJO) complexed with the Erlotinib inhibitor (PDB code AQ4), and an MD simulation of the complex formed by the SARS-CoV-2 Receptor Binding Domain and the human Angiotensin

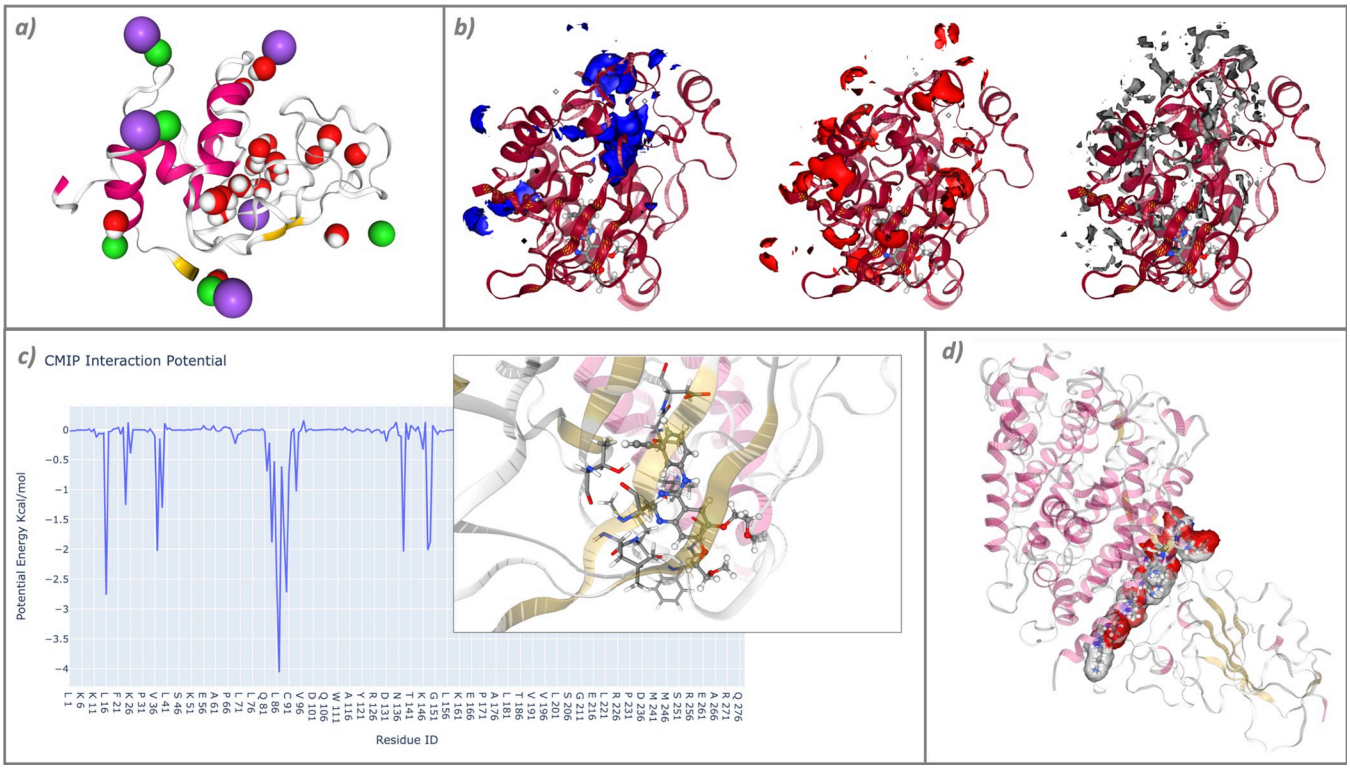

Fig 6. Intermediate results extracted from the "Molecular Interaction Potentials" workflow. (a) Structural water molecules and ions placed in the energetically most favorable spots on the surface of the protein; (b) molecular interaction potential grids obtained from a positive probe (left, blue), a negative probe (middle, red), and a neutral probe (right, gray); and (c) potential energy (electrostatic + VdW) calculated for a protein–ligand interaction. The inline plot shows the representation of the protein residues with lower energy (higher affinity). (d) Representation of the residues contributing the most to the protein–protein complex interaction.

https://doi.org/10.1371/journal.pcbi.1012173.g006

Converting Enzyme 2 (PDB code 6VW1). The software tool wrapped by the BioBB library and used in this workflow is called Classical Molecular Interaction Potentials (CMIP) [46].

The workflow is divided into 3 main sections: structural water molecules and ions (titration), MIPs, and interaction potential energies. In the first part, the pipeline shows how to place a desired number of water molecules and ions in the energetically most favorable spots on the surface of the protein, a useful method for the preparation of input systems for MD simulations (Fig 6A). In the second part, molecular interaction potential grids are generated with 3 distinct electro-charged probes: positive, negative, and neutral. The representation of the resulting grids allows graphical exploration of regions of the proteins with affinity for negative, positive, or hydrophobic residues (respectively) (Fig 6B). Finally, in the third part, examples of how to compute molecular interaction energies in protein–ligand and protein–protein complexes are presented. The method uses the Poisson–Boltzmann equation to calculate the electrostatic potential by default, although all parameters can be tuned. The results are parsed and energies accumulated by residue, thereby allowing for easy identification of the residues with the greatest contribution to the interaction (Fig 6C). The interaction region can also be easily identified and represented with a simple energy cutoff (Fig 6D).

Discussion/Conclusions

The FAIRification of computational biomolecular simulation workflows following the principles presented by initiatives like FAIR4RS [10] is starting to promote reproducibility and

reusability in the field. We believe that the collection of workflows presented herein is a good example. They fulfill the most important rules for FAIR workflow implementation [47]: workflows are properly registered and described with rich metadata (Findability); source code is available in public code repositories and example input data and results are provided along with the workflows (Accessibility); tools integrated into the workflows are interoperable between them thanks to the BioBB library (see Methods) and are adhering to file format standards (Interoperability); and finally they come with an associated reproducible computational environment to run the workflow, are flexible with parameter tuning and compatible with input configuration files, are highly modular (BioBB building blocks), with the possibility to easily swap building blocks and reconfigure the workflow, contain clear and concise documentation included as markdown cells, and include qualified references to the software used through a list of citations (Reusability/Reproducibility). All these points are explained in detail in the Materials and methods section.

Two of these FAIR points are especially relevant for our field: the possibility to extensively document the workflow and the high degree of reusability/reproducibility offered by the packaging systems used. The combination of these led to a collection of workflows prepared to be used in training sessions and hands-on tutorials, with students acknowledging: (i) not having to deal with any software dependency installation; (ii) having detailed associated documentation for each of the steps of the workflow; (iii) interactively working with the workflow; (iv) graphically inspecting the intermediate results; and (v) having the possibility to modify/expand an existing workflow.

The possibility to use myBinder to automatically install and execute Jupyter Notebooks in the cloud is highly convenient for the free deployment of these types of demonstration workflows, as it offers the interactive execution of the workflow with just 1 click. The possibility of installing myBinder engine (BinderHub) locally in a private cloud environment opens the door to personalized Binders, providing more control over the number of computational resources and simultaneous users allowed. Additionally, these technologies are starting to reach HPC centers, connecting the power of supercomputers directly from Jupyter Notebooks, a very important step for our kind of calculations. Finally, similar infrastructures are also provided by the most important cloud providers in the world (Amazon, Google, Microsoft's Azure), offering some interesting advantages such as the possibility to access GPU cards. However, most of these infrastructures are not compatible with the use of Conda environments, which means that additional packages and software dependencies need to be installed inside the workflow and reinstalled at the start of every session, thus hindering reproducibility aspects.

Finally, the presented workflows are transversal, common, and useful pipelines that can be applied as sub-workflows in more complex studies or different platforms. In fact, these demonstration workflows were converted, when possible, to different formats, thanks to the compatibility of the BioBB library with a range of workflow managers [19]. Common Workflow Language (CWL) [48], Galaxy [49], and pure Python versions of the presented workflows are available from a centralized GitHub repository (https://github.com/bioexcel/biobb_workflows/). From these, pure Python scripts, in combination with HPC workflow managers such as PyCOMPSs [50,51], were used in real scientific cases [52,53]. We would like to stress here that, in contrast to these last examples used in HPC environments, the notebooks presented in this work are designed for educational purposes only and not for production usage.

We believe that FAIR and reproducible workflows are the right way to build, execute, and share our scientific pipelines, and we will continue to expand our collection of workflows in the coming years.

Materials and methods

The digital notebooks presented in this work are strongly reliant on Jupyter Notebooks [54] and Python programming language. The BioBB library [17,19] is the main engine behind the workflows. BioPython library [55] is heavily used in the BioBB modules to process and work with PDB files. NGLview [42], py3Dmol [56], and simpletraj (https://github.com/arose/simpletraj) are used to interactively visualize 3D structures and MD trajectories. Plotly [44] and matplotlib [57] are used to display and plot data with 2D graphs.

Conda (https://docs.conda.io/) is used as a software packaging system. All BioBB modules and associated dependencies (when needed) are packaged with Conda. Conda environments (https://www.anaconda.com/) allow us to make workflows easily reproducible in different machines, using packages from BioConda [18] and conda-forge community channels. Of note, although Conda environments are compatible and can be used with all kinds of Operative Systems (OS), including Apple, Linux, and Windows, the tools used in the workflow (dependencies) are those that will finally determine this compatibility. As an example, if a workflow uses a tool that is compatible only with Linux, the whole workflow will then be compatible only with this specific OS.

MyBinder (https://mybinder.org/) is the technology used to offer the notebooks in an executable and reproducible environment.

GitHub is the chosen repository and version control system to store the digital notebooks. The BioExcel repository (https://github.com/bioexcel) is used to concentrate the collection of BioBB workflows. Each workflow repository contains an associated Conda virtual environment configuration file, including all the needed dependencies that will be automatically installed in the Conda environment. All workflows are also available from the WorkflowHub registry [58], the ELIXIR bioinformatics registry bio.tools [59], and from a central repository with all the available versions of the workflows (Jupyter Notebooks, Common Workflow Language (CWL) [48], pure Python and Galaxy [49]), along with the corresponding installation instructions: https://github.com/bioexcel/biobb_workflows/.

Read the docs (https://docs.readthedocs.io/) is the technology chosen to document the workflows. All workflows contain an Introduction and installation section and a Tutorial section. All input and output files are linked to their corresponding EDAM ontology [60] (e.g., AMBER parmtop—edam:format_3881). These EDAM terms are also used in the bio.tools entries and CWL specifications.

JavaScript Object Notation for Linking Data (JSON-LD, https://github.com/json-ld/json-ld.org) is used to capture the software references for the BioBB workflows, enclosing a list of citations to all the software that is used in the pipeline. JSON-LD format is a standard-based machine-readable data, in addition to be easy for humans to read and write.

The environmental footprint for a typical run of the presented workflows (1:30 h on 12 CPUs Xeon E5-2683 v4) draws 170.76 Wh. Based in Spain, this has a carbon footprint of 29.21g CO2e, which is equivalent to 0.03 tree-months (calculated using green-algorithms.org v2.2 [61]). Scaling up these workflows to production runs considerably increases the environmental footprint, varying significantly depending on the HPC resources used.

The installation process is shared by all the workflows included in the collection and is based on 4 easy steps:

1. Cloning the GitHub repository (*git clone*).
2. Creating the Conda environment with all required workflow dependencies (*conda env create*). A specific yaml-formatted file is offered for each workflow with the list of required dependencies (tools and libraries).

3. Activating the environment (*conda activate*).
4. Deploying the Jupyter Notebook (*jupyter-notebook*).

Alternatively, a link to myBinder is offered, with a direct deployment, thus avoiding the installation process.

The collection of notebooks is available here:
https://mmb.irbbarcelona.org/biobb/workflows.

Additional links (GitHub, WorkflowHub, bio.tools, and DOIs) are included in S1 Data.

Supporting information

S1 Data. Table A. Collection of FAIR BioBB biomolecular simulation workflows implemented in Jupyter Notebooks. Fig A. Uniform header for all the BioBB tutorials, exemplified with the GROMACS Protein MD setup workflow. Sections included are: (i) title and description; (ii) BioBB modules used; (iii) auxiliary libraries used; (iv) command lines required to install and launch the workflow; and (v) pipeline steps.
(DOCX)

Author Contributions

Conceptualization: Josep Lluís Gelpí, Adam Hospital.

Funding acquisition: Modesto Orozco.

Methodology: Genís Bayarri, Josep Lluís Gelpí, Adam Hospital.

Software: Genís Bayarri, Pau Andrio, Adam Hospital.

Supervision: Adam Hospital, Modesto Orozco.

Validation: Genís Bayarri, Pau Andrio, Adam Hospital.

Writing – original draft: Adam Hospital.

Writing – review & editing: Genís Bayarri, Adam Hospital.

References

1. Munafò MR, Nosek BA, Bishop DVM, Button KS, Chambers CD, du Sert NP, et al. A manifesto for reproducible science. Nat Hum Behav. 2017; 1 (1):0021. https://doi.org/10.1038/s41562-016-0021 PMID: 33954258
2. Coveney PV, Groen D, Hoekstra AG. Reliability and reproducibility in computational science: implementing validation, verification and uncertainty quantification in silico. Philos Trans R Soc A Math Phys Eng Sci. 2021; 379(2197):20200409. https://doi.org/10.1098/rsta.2020.0409 PMID: 33775138
3. Peng RD. Reproducible Research in Computational Science. Science. 2011; 334(6060):1226–1227. https://doi.org/10.1126/science.1213847 PMID: 22144613
4. Schubotz M, Satpute A, Greiner-Petter A, Aizawa A, Gipp B. "Caching and Reproducibility: Making Data Science Experiments Faster and FAIRer," Frontiers in Research Metrics and Analytics. Perspective. 2022;7. https://doi.org/10.3389/frma.2022.861944 PMID: 35531060
5. Lewis J, Breeze CE, Charlesworth J, Maclaren OJ, Cooper J. Where next for the reproducibility agenda in computational biology? BMC Syst Biol. 2016; 10(1):52. https://doi.org/10.1186/s12918-016-0288-x PMID: 27422148
6. Thompson MW, Gilmer JB, Matsumoto RA, Quach CD, Shamaprasad P, Yang AH, et al. Towards molecular simulations that are transparent, reproducible, usable by others, and extensible (TRUE). Mol Phys. 2020; 118(9–10):e1742938. https://doi.org/10.1080/00268976.2020.1742938 PMID: 33100401
7. Grüning B, Chilton J, Köster J, Dale R, Soranzo N, van den Beek M, et al. Practical Computational Reproducibility in the Life Sciences. Cell Syst. 2018; 6(6):631–635. https://doi.org/10.1016/j.cels.2018.03.014 PMID: 29953862

8. Schaduangrat N, Lampa S, Simeon S, Gleeson MP, Spjuth O, Nantasenamat C. Towards reproducible computational drug discovery. J Cheminf. 2020; 12(1):9. https://doi.org/10.1186/s13321-020-0408-x PMID: 33430992
9. Wilkinson MD, Dumontier M, Aalbersberg IJ, Appleton G, Axton M, Baak A, et al. The FAIR Guiding Principles for scientific data management and stewardship. Commentary. 2016; 3:160018. https://doi.org/10.1038/sdata.2016.18 PMID: 26978244
10. Barker M, Chue HN, Katz DS, Lamprecht AL, Martinez-Ortiz C, Psomopoulos F, et al. "Introducing the FAIR Principles for research software," (in eng). Sci Data. 2022; 9 (1):622. https://doi.org/10.1038/s41597-022-01710-x PMID: 36241754
11. del Pico EM, Gelpí JL, Capella-Gutiérrez S "FAIRsoft—A practical implementation of FAIR principles for research software," bioRxiv, p. 2022.05.04.490563, 2022. https://doi.org/10.1101/2022.05.04.490563
12. Lamprecht A-L, Garcia L, Kuzak M, Martinez C, Arcila R, Martin E, et al., "Towards FAIR principles for research software." Data Sci. 2020; 3(1):37–59. https://doi.org/10.3233/DS-190026
13. Hasselbring W, Carr L, Hettrick S, Packer H, Tiropanis T. From FAIR research data toward FAIR and open research software. 2020; 62(1):39–47. https://doi.org/10.1515/itit-2019-0040
14. Abraham M, Apostolov R, Barnoud J, Bauer P, Blau C, Bonvin AMJJ, et al. Sharing Data from Molecular Simulations. J Chem Inf Model. 2019; 59(10):4093–4099. https://doi.org/10.1021/acs.jcim.9b00665 PMID: 31525920
15. Goñi R, Apostolov R, Lundborg M, Bernau C, Jamitzky F, Laure E, et al. Standards for data handling Zenodo. 2013. https://doi.org/10.5281/zenodo.4468929
16. Elofsson A, Hess B, Lindahl E, Onufriev A, van der Spoel D, Wallqvist A. Ten simple rules on how to create open access and reproducible molecular simulations of biological systems. PLoS Comput Biol. 2019; 15(1):e1006649. https://doi.org/10.1371/journal.pcbi.1006649 PMID: 30653494
17. Andrio P, Hospital A, Conejero J, Jordá L, Del Pino M, Codo L, et al. "BioExcel Building Blocks, a software library for interoperable biomolecular simulation workflows." Sci Data. 2019; 6(1):169. https://doi.org/10.1038/s41597-019-0177-4 PMID: 31506435
18. Grüning B, Dale R, Sjödin A, Chapman BA, Rowe J, Tomkins-Tinch CH, et al. "Bioconda: sustainable and comprehensive software distribution for the life sciences." Nat Methods., vol. 15, no. 7, p. 475–476, 07 2018. https://doi.org/10.1038/s41592-018-0046-7 PMID: 29967506
19. Soiland-Reyes S, Bayarri G, Andrio P, Long R, Lowe D, Niewielska A, et al. Making Canonical Workflow Building Blocks Interoperable across Workflow Languages. Data Intelligence. 2022; 4(2):342–357. https://doi.org/10.1162/dint_a_00135
20. Lasser J. Creating an executable paper is a journey through Open Science. Commun Phys. 2020; 3:1–5. https://doi.org/10.1038/s42005-020-00403-4
21. Figueiredo L, Scherer C, Cabral JS. A simple kit to use computational notebooks for more openness, reproducibility, and productivity in research. PLoS Comput Biol. 2022; 18(9):e1010356. https://doi.org/10.1371/journal.pcbi.1010356 PMID: 36107931
22. Richardson R, Çelebi R, van der Burg S, Smits D, Ridder L, Dumontier M, et al. "User-friendly Composition of FAIR Workflows in a Notebook Environment," presented at the Proceedings of the 11th Knowledge Capture Conference, Virtual Event, USA, 2021. [Online]. https://doi.org/10.1145/3460210.3493546
23. Davies A, Hooley F, Causey-Freeman P, Eleftheriou I, Moulton G. Using interactive digital notebooks for bioscience and informatics education. PLoS Comput Biol. 2020; 16(11):e1008326. https://doi.org/10.1371/journal.pcbi.1008326 PMID: 33151926
24. Gupta YM, Kirana SN, Homchan S, Tanasarnpaiboon S. Teaching Python programming for bioinformatics with Jupyter notebook in the Post-COVID-19 era. Biochem Mol Biol Educ. 2023; 51(5). https://doi.org/10.1002/bmb.21746 PMID: 37199252
25. Centeno EGZ, Moreni G, Vriend C, Douw L, Santos FAN. A hands-on tutorial on network and topological neuroscience. Brain Struct Funct. 2022; 227 (3):741–762. https://doi.org/10.1007/s00429-021-02435-0 PMID: 35142909
26. Le KH, Adolf-Bryfogle J, Klima JC, Lyskov S, Labonte J, Bertolani S, et al. PyRosetta Jupyter Notebooks Teach Biomolecular Structure Prediction and Design. Biophysicist. 2021; 2(1):108–122. https://doi.org/10.35459/tbp.2019.000147 PMID: 35128343
27. Anon, 2020. Anaconda Software Distribution, Anaconda Inc. Available from: https://docs.anaconda.com/.
28. Bayarri G, Andrio P, Hospital A, Orozco M, Gelpí JL. BioExcel Building Blocks Workflows (BioBB-Wfs), an integrated web-based platform for biomolecular simulations. Nucleic Acids Res. 2022; 50(W1):W99–W107. https://doi.org/10.1093/nar/gkac380 PMID: 35639735

29. BioBB_workflows. "Protein MD Setup tutorial using BioExcel Building Blocks (BioBB)." BioExcel CoE. https://doi.org/10.48546/workflowhub.workflow.120.6
30. Abraham MJ, Murtola T, Schulz R, Páll S, Smith JC, Hess B, et al. GROMACS: High performance molecular simulations through multi-level parallelism from laptops to supercomputers. SoftwareX. 2015;1–2:19–25. https://doi.org/10.1016/j.softx.2015.06.001
31. Lemkul JA. "From Proteins to Perturbed Hamiltonians: A Suite of Tutorials for the GROMACS-2018 Molecular Simulation Package [Article v1.0]." Living J Comput Mol Sci. 2018; 1(1):5068. https://doi.org/10.33011/livecoms.1.1.5068
32. Berman HM, Westbrook J, Feng Z, Gilliland G, Bhat TN, Weissig H, et al. The Protein Data Bank. Nucleic Acids Res. 2000; 28(1):235–242. https://doi.org/10.1093/nar/28.1.235 PMID: 10592235
33. BioBB_workflows. "Protein-ligand Docking tutorial using BioExcel Building Blocks (BioBB)." BioExcel CoE. https://doi.org/10.48546/workflowhub.workflow.129.5
34. Law V, Knox C, Djoumbou Y, Jewison T, Guo AC, Liu Y, et al. "DrugBank 4.0: shedding new light on drug metabolism." Nucleic Acids Res. vol. 42, no. Database issue, p. D1091–7, Jan 2014. https://doi.org/10.1093/nar/gkt1068 PMID: 24203711
35. Le Guilloux V, Schmidtke P, Tuffery P. Fpocket: An open source platform for ligand pocket detection. BMC Bioinformatics. 2009; 10(1):168. https://doi.org/10.1186/1471-2105-10-168 PMID: 19486540
36. Trott O, Olson AJ. AutoDock Vina: improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading. J Comput Chem. 2010; 31(2):455–461. https://doi.org/10.1002/jcc.21334 PMID: 19499576
37. BioBB_workflows. "Protein structure flexibility tutorial using BioExcel Building Blocks (BioBB) and FlexServ tools." BioExcel CoE. https://doi.org/10.48546/workflowhub.workflow.551.2
38. Camps J, Carrillo O, Emperador A, Orellana L, Hospital A, Rueda M, et al. FlexServ: an integrated tool for the analysis of protein flexibility. Bioinformatics. 2009;1; 25(13):1709–10. https://doi.org/10.1093/bioinformatics/btp304 PMID: 19429600
39. Meyer T, Ferrer-Costa C, Pérez A, Rueda M, Bidon-Chanal A, Luque FJ, et al. Essential Dynamics: A Tool for Efficient Trajectory Compression and Management. J Chem Theory Comput. 2006; 2(2):251–258. https://doi.org/10.1021/ct050285b PMID: 26626512
40. Shkurti A, Goñi R, Andrio P, Breitmoser E, Bethune I, Orozco M, et al. pyPcazip: A PCA-based toolkit for compression and analysis of molecular simulation data. SoftwareX. 2016; 5:44–50. https://doi.org/10.1016/j.softx.2016.04.002
41. Amadei A, Linssen AB, Berendsen HJ. Essential dynamics of proteins. Proteins. 1993; 17(4):412–425. https://doi.org/10.1002/prot.340170408 PMID: 8108382
42. Nguyen H, Case DA, Rose AS. NGLview-interactive molecular graphics for Jupyter notebooks. Bioinformatics. 2018; 34(7):1241–1242. https://doi.org/10.1093/bioinformatics/btx789 PMID: 29236954
43. Rose AS, Bradley AR, Valasatava Y, Duarte JM, Prlić A, Rose PW. NGL viewer: web-based molecular graphics for large complexes. Bioinformatics. 2018:pp. bty419–bty419. https://doi.org/10.1093/bioinformatics/bty419 PMID: 29850778
44. Plotly Technologies Inc. Title: Collaborative data science Publisher: Plotly Technologies Inc. Place of publication: Montréal, QC Date of publication: 2015 URL: https://plot.ly.
45. BioBB_workflows. "CMIP tutorial using BioExcel Building Blocks (BioBB)." BioExcel CoE. https://doi.org/10.48546/workflowhub.workflow.773.2.
46. Gelpí JL, Kalko SG, Barril X, Cirera J, de La Cruz X, Luque FJ, et al. Classical molecular interaction potentials: improved setup procedure in molecular dynamics simulations of proteins. Proteins. vol. 45, no. 4, p. 428–37, Dec 2001. https://doi.org/10.1002/prot.1159 PMID: 11746690
47. de Visser C, Johansson LF, Kulkarni P, Mei H, Neerincx P, van der Velde KJ, et al. Ten quick tips for building FAIR workflows. PLoS Comput Biol. 2023; 19(9):e1011369. https://doi.org/10.1371/journal.pcbi.1011369 PMID: 37768885
48. Amstutz P., Crusoe M. R., Tijanić N., Chapman B., Chilton J., Heuer M., et al., Common Workflow Language, v1.0. 2016
49. Afgan E, Baker D, van den Beek M, Blankenberg D, Bouvier D, Čech M, et al. The Galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2016 update. Nucleic Acids Res. 2016; 44(W1):W3–W10. https://doi.org/10.1093/nar/gkw343 PMID: 27137889
50. Tejedor E, Becerra Y, Alomar G, Queralt A, Badia RM, Torres J, et al. PyCOMPSs: Parallel computational workflows in Python. Int J High Perform Comput Appl. 2017; 31(1):66–82. https://doi.org/10.1177/1094342015594678
51. Ejarque J, Andrio P, Hospital A, Conejero J, Lezzi D, Gelpí JL, et al. The BioExcel methodology for developing dynamic, scalable, reliable and portable computational biomolecular workflows. In 2022

IEEE 18th International Conference on e-Science (e-Science), 2022, p. 357–366. https://doi.org/10.1109/eScience55777.2022.00049

52. Wieczór M, Genna V, Aranda J, Badia RM, Gelpí JL, Gapsys V, et al. Pre-exascale HPC approaches for molecular dynamics simulations. Covid-19 research: A use case. WIREs Comput Mol Sci. 2023; 13 (1):e1622. https://doi.org/10.1002/wcms.1622 PMID: 35935573

53. Suriñach A, Hospital A, Westermaier Y, Jordà L, Orozco-Ruiz S, Beltrán D, et al. High-Throughput Prediction of the Impact of Genetic Variability on Drug Sensitivity and Resistance Patterns for Clinically Relevant Epidermal Growth Factor Receptor Mutations from Atomistic Simulations. J Chem Inf Model. 2023; 63(1):321–334. https://doi.org/10.1021/acs.jcim.2c01344 PMID: 36576351

54. Kluyver T, Ragan-Kelley B, Pérez F, Granger B, Bussonnier M, Frederic J, et al. Jupyter Notebooks—a publishing format for reproducible computational workflows. Presented at the Positioning and Power in Academic Publishing: Players, Agents and Agendas, 2016. https://doi.org/10.3233/978-1-61499-649-1-87

55. Cock PJ, Antao T, Chang JT, Chapman BA, Cox CJ, Dalke A, et al. Biopython: freely available Python tools for computational molecular biology and bioinformatics. Bioinformatics. 2009; 25(11):1422–1423. https://doi.org/10.1093/bioinformatics/btp163 PMID: 19304878

56. Rego N, Koes D. 3Dmol.js: molecular visualization with WebGL. Bioinformatics. 2015; 31(8):1322–1324. https://doi.org/10.1093/bioinformatics/btu829 PMID: 25505090

57. Hunter JD. Matplotlib: A 2D Graphics Environment. Comput Sci Eng. 2007; 9(3):90–95. https://doi.org/10.1109/MCSE.2007.55

58. Goble C, Soiland-Reyes S, Bacall F, Owen S, Williams A, Eguinoa I, et al. Implementing FAIR Digital Objects in the EOSC-Life Workflow Collaboratory. 2021. Zenodo. https://doi.org/10.5281/zenodo.4605654

59. Ison J, Rapacki K, Ménager H, Kalaš M, Rydza E, Chmura P, et al. Tools and data services registry: a community effort to document bioinformatics resources. Nucleic Acids Res. 2016; 44(D1):D38–D47. https://doi.org/10.1093/nar/gkv1116 PMID: 26538599

60. Ison J, Kalas M, Jonassen I, Bolser D, Uludag M, McWilliam H, et al. EDAM: an ontology of bioinformatics operations, types of data and identifiers, topics and formats. Bioinformatics. 2013; 29(10):1325–1332. https://doi.org/10.1093/bioinformatics/btt113 PMID: 23479348

61. Lannelongue L, Grealey J, Inouye M. Green Algorithms: Quantifying the Carbon Footprint of Computation. Adv Sci (Weinh). 2021; 8(12):2100707. https://doi.org/10.1002/advs.202100707 PMID: 34194954