INFRAEDI-02-2018

Center of Excellence for Computational Biomolecular Research

www.bioexcel.eu

BioExcel-2 Project Number 823830

D2.2 – First release of workflow-ready building blocks library

WP2: Convergence of HPC/HPDA and Improved Usability

Document Information

Deliverable Number	D2.2
Deliverable Name	First release of workflow-ready building blocks library
Due Date	2019-12-31 (PM12)
Deliverable Lead	IRB
Authors	Adam Hospital (IRB), Genís Bayarri (IRB), Stian Soiland-Reyes (UNIMAN), Josep Lluís Gelpí (BSC), Pau Andrio (BSC), Daniele Lezzi (BSC), Sarah Butcher (EMBL-EBI), Ania Niewielska (EMBL-EBI), Yvonne Westermaier (NBD), Rosa Maria Badia (BSC)
Keywords	Building Blocks, Interoperability, FAIR, Tools, Workflows, Workflow managers, Hardware infrastructures, Container technologies
WP	WP2
Nature	Report
Dissemination Level	Public
Final Version Date	2019-12-23
Reviewed by	PMB
MGT Board Approval	2019-12-27

Document History

Partner	Date	Comments	Version
IRB	2019-11-04	Initial draft	0.1
IRB	2019-11-24	First complete draft	0.2
IRB, NBD, BSC	2019-12-03	First internal review	0.3
IRB, MPG, UU, KTH	2019-12-19	EB review comments addressed	0.4
IRB	2019-12-23	Final draft for PMB approval	0.5
UNIMAN	2021-03-15	DOIs	0.5.1

Executive Summary

This deliverable presents the first release of the workflow-ready BioExcel building blocks library for the BioExcel-2 period. The initial roadmap of the novel identified building blocks presented in D2.1 is revised, and a new, updated roadmap for the rest of the project is presented.

The current situation section contains a description of the new release of the BioExcel building blocks library, including the content of the release, the new library modules recently developed, updates on the already existing modules, new functionalities and bug fixes, and work done into increasing its visibility and outreach.

The initial and updated roadmap section contains a revision of the initial roadmap presented in the D2.1 document in PM6, highlighting the milestones reached and issues found. From that, an updated roadmap for the development of the software library during the next 2 years is proposed, including a large set of new building blocks.

Contents

1 INTRODUCTION 6

2 CURRENT SITUATION 8

2.1 LIBRARY RELEASE 8
2.1.1 SOURCE CODE & PACKAGES 8
2.1.2 DOCUMENTATION 10
2.2 NEW CATEGORIES 11
2.2.1 CHEMISTRY MODULE 11
2.2.2 STRUCTURE UTILS MODULE 12
2.2.3 PMX MODULE 14
2.3 UPDATED CATEGORIES 14
2.3.1 IO MODULE 14
2.3.2 MD MODULE 15
2.3.3 ANALYSIS MODULE 16
2.4 NEW FUNCTIONALITIES 18
2.4.1 CONTAINER EXECUTION COMPATIBILITY 18
2.4.2 WORKFLOW PARAMETERS 19
2.4.3 BUILDING BLOCKS COMMON PARAMETERS 20
2.5 BUG FIXES 20
2.6 VISIBILITY AND OUTREACH 21
2.6.1 LANDING WEBSITE 22
2.6.2 JUPYTER NOTEBOOKS TUTORIALS 23

3 INITIAL & UPDATED ROADMAP 24

3.1 INITIAL ROADMAP 25
3.2 UPDATED ROADMAP 26

4 CONCLUSIONS 30

5 REFERENCES 31

1 Introduction

One of the main pillars of the BioExcel Centre of Excellence since the beginning of the project 4 years ago has been helping our community with the design and development of reproducible computational biomolecular workflows tackling real scientific problems. The first step towards this goal was to try to build demonstration workflows including the main software packages being developed and maintained by our partners in the consortium: GROMACS, HADDOCK, CPMD/CP2K and pmx. Already at this initial stage, one of the most well-known issues in the workflows field appeared: software interoperability. Building workflows interconnecting our main codes required input and output data compatibility. This was the main motivation behind the BioExcel building blocks (*biobb*), a library of Python wrappers offering a new layer of compatibility and interoperability over the popular BioExcel computational biomolecular tools.

During the initial design of our *biobb* software library, BioExcel adopted the objective of pushing the ELIXIR bioinformatics' concept and usage of workflows into the biomolecular research field, combining ELIXIR's recommendations and services with biomolecular simulation, thus demonstrating the feasibility of working according to the FAIR principles in this field. The FAIR guiding principles [1] of data management put specific emphasis on enhancing the ability of machines to automatically find and use the data, in addition to supporting its reuse by individuals. OSS recommendations [2] and a very recent initiative presented in a position paper [3], on the other hand, encourage the adoption of existing best practices and FAIR principles in the field of research software development. BioExcel's building blocks library is being designed following these FAIR principles applied to software. The result of the first version released in the BioExcel-1 period was a fully interoperable software library primarily based on the GROMACS MD package, with workflows built using such components being executed in a set of popular workflow managers and middleware, and in a number of complementary computational environments. Besides, the components were described using CWL [6] and OpenAPI, to improve access and portability, with the added possibility to run them in CWL-compliant workflow managers. The library was published in the *Nature Scientific Data* [7] in September 2019.

A pre-exascale study launched in 40,000 cores of the Marenostrum BSC supercomputer was successfully run to prove the exascale possibilities of the library coupled to the PyCOMPSs workflow manager [4]. Results of the execution, consisting on 200 Molecular Dynamics simulations for a selected collection of annotated variants, were used to extract a set of descriptors that are currently being integrated in a pathogenicity predictor (PMut [5]), studying the value of protein dynamics information to predict variant pathogenicity. Following this biobb+PyCOMPSs approach, new massively parallel studies are being prepared in the context of the *rational drug design* use case. Using the new building blocks presented in this document, ligand binding free energies will be calculated for a particular protein system and a set of variants, to study the impact of these mutations to the ligand binding affinity. The possibility to use PyCOMPSs workflow manager to control a workflow built using the BioExcel building blocks library allows an efficient use of up to hundreds of thousands of cores, thanks to its automatic management of data dependencies and potential parallelization of a simple Python code.

After this 1st phase, the clear goal for the BioExcel-2 period is to increase the number of tools wrapped using the *biobb* library. The initial set of building blocks wrapping mainly GROMACS tools will be extended to include the main BioExcel codes, with the goal of using them to build complex workflows helping in solving real scientific questions (e.g. BioExcel use cases). An initial roadmap for further development of the library during the first year of the BioExcel-2 period, coupled to the Task 2.1 (*Application building blocks for computational biomolecular simulations*) was presented in BioExcel-2 D2.1. Milestones presented in this initial roadmap for the first year of the project have been reached in mostly all objectives. Only one objective has been identified as being delayed (data analytics building blocks), and will be studied and tackled appropriately in the coming months. In this document, the new features included in the current library release (Dec. 2019) are described, the initial roadmap is revised, and an updated roadmap for the next 2 years is proposed.

2 Current Situation

New and updated categories, functionalities and bug fixes included in the December 2019 release are presented in the next sections. The main functionalities added are the chemistry, structure_utils and pmx modules, which allow the addition of small molecule format conversions and automatic generation of topologies (chemistry), information extraction from PDB files (structure_utils) and free energy calculations (pmx) to the *biobb* workflows. Updates in already existing *biobb* categories, bug fixes, and work done towards increasing the library visibility and extending its outreach are also briefly described.

2.1 Library Release

The first official release of the workflow-ready BioExcel building blocks library for the BioExcel-2 project has been launched in December 2019 (v2.0.0, 2019.4 release). Following the steps of the main software codes in BioExcel, 4 main releases are planned per year, i.e. one every 3 months. The release contains two important sections: 1) source code, packages and containers; and 2) their associated documentation.

2.1.1 Source code & packages

The biobb software library and all its related workflows are presented under the Apache License 2.0, see the file LICENSE for details. This is a *permissive open source* license allowing redistribution, reuse and repurposing, as long as the attributions and original license are retained. However, underlying software, which these wrappers effectively call as command line tools, are distributed under several open source licenses.

The library, following the **FAIR** software development principles [3] as described in previous deliverables, is easily findable and accessible from multiple sites and with multiple packaging methods:

- **bio.tools** (https://bio.tools/biobb): The library is registered as a single entry in the European Bioinformatics Software Registry bio.tools of the ELIXIR initiative. The entry includes associated software metadata information about the package such as links to all the modules (GitHub repositories, BioConda packages), documentation, contact details, publications, training material, etc.

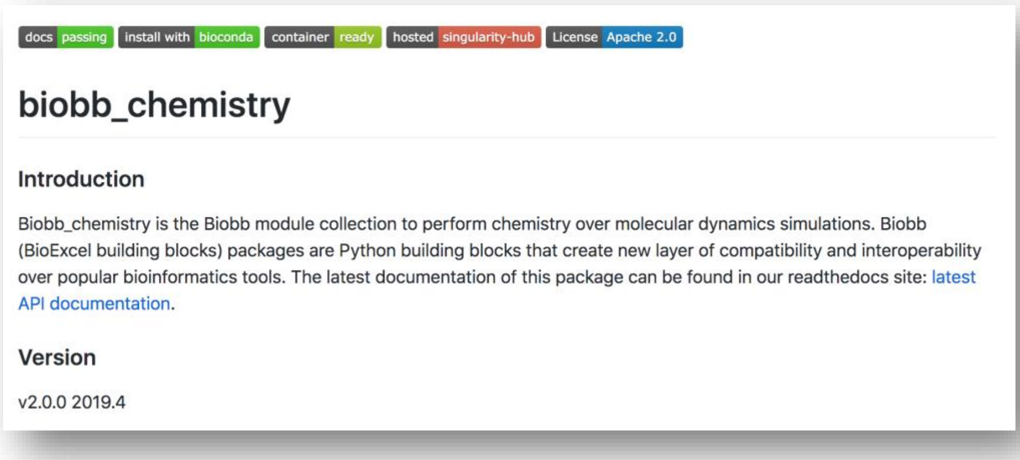
- **GitHub** (https://github.com/bioexcel/biobb): GitHub is the main repository for the development of the library. Every module of the package has its own repository (e.g. md, chemistry, pmx), including the source code and links (as badges) to the corresponding API documentation, Conda package, containers (Docker, Singularity) and license (Fig. 1).

- **BioConda** (https://anaconda.org/search?q=biobb): Every module of the library is bundled as a Conda package and made available through the

BioConda channel. This allows an easy installation process, as all the software dependencies are described in the Conda package and tackled by the Conda manager.

- **Docker - BioContainers** (https://biocontainers.pro/#/registry): Every Conda package of the library is converted to a Docker container and made available through the BioContainers registry. This allows a direct execution of the library wrappers, without any additional software installation other than the Docker engine. This is particularly interesting in biomolecular workflows combining many different software tools.

- **Singularity** (https://singularity-hub.org/): Every Docker container of the library is converted into a Singularity container and made available through the singularity-hub registry. This allows the use of the library containers in HPC centers, which prefer Singularity to Docker technology due to security aspects.



biobb_chemistry

Introduction

Biobb_chemistry is the Biobb module collection to perform chemistry over molecular dynamics simulations. Biobb (BioExcel building blocks) packages are Python building blocks that create new layer of compatibility and interoperability over popular bioinformatics tools. The latest documentation of this package can be found in our readthedocs site: latest API documentation.

Version

v2.0.0 2019.4

Figure 1.- Screenshot of the biobb_chemistry module GitHub landing page as an example. Badges available for every module (top) correspond to documentation (readthedocs), BioConda package, Docker/Singularity containers and Apache License.

The software library, as described in previous deliverables, is divided into categories according to the functionality of the tools wrapped. The list of the categories available in the current release is:

- **biobb_common**: base package required to use the biobb library.
- **biobb_io**: collection to fetch data from biological databases.
- **biobb_model**: collection to check and model 3D structures.
- **biobb_md**: collection to perform Molecular Dynamics simulations.
- **biobb_analysis**: collection to perform analysis of MD simulations.
- **biobb_pmx**: collection to perform free energy calculations.
- **biobb_structure_utils**: collection to extract information from a PDB structure file.
- **biobb_chemistry**: collection to perform cheminformatics analyses and format conversions.

All together, the library offers more than 50 different wrappers on top of popular biomolecular software tools, with an interoperable layer that allows their direct interconnection, giving the possibility to build complex biomolecular workflows. Every building block contains an associated unit testing code, which can be proved with the Python unittest unit-testing framework.

A summary of the software library categories and associated building blocks, together with links to their documentation and all the available Conda packages and Docker/Singularity containers is accessible from a new web site developed as an entry point to present the library to the biomolecular simulation community: http://mmb.irbbarcelona.org/biobb/ (see section 2.6.1). The list of building blocks presented on the web site is automatically updated: it is connected to the GitHub repositories, so any time a new commit arrives, an update of the list is triggered (GitHub webhooks), ensuring coherence between what is available and what is presented.

2.1.2 Documentation

All the building blocks from the library are documented using readthedocs, and described using CWL [6] and openAPI.

Documentation is automatically generated from the comments included in the code following docstrings conventions, transformed afterwards to html and markdown using sphinx. Every biobb category has its own readthedocs documentation, divided in three main sections:

- **Introduction & installation**: containing information on how to install and use the building blocks for the particular module.
- **API Documentation**: containing extensive information on every building block (wrapper), including input parameters and properties definition, with data types and default values.
- **Command Line Documentation**: containing information on how to run the building blocks in command-line, as a binary, executable program.

Links to all the readthedocs documentation pages can be found in the central biobb GitHub repository and also in the library website.

CWL descriptions for all the building blocks are available in the biobb adapters module. The descriptions are linked to the corresponding Docker container generated for the biobb module, which allows using them directly with any CWL-compliant workflow manager, such as cwltool or toil [8]. For extended information including examples, please refer to previous deliverables [9].

JSON schemas for inputs, outputs and properties parameters for each building block are available in the GitHub repository of every biobb module (e.g.

biobb chemistry). The collection of schemas, together with a central schema describing the endpoints corresponding to every available wrapper in the module, generates a complete OpenAPI specification. This specification can then be used to generate a REST API server with interactive API documentation, letting the users to try out the API calls directly in the browser (work in progress, see updated roadmap section).

2.2 New categories

The BioExcel building blocks library has been significantly extended since the first version presented in the first period of BioExcel, described in detail in the publication by *Andrio et al.* [7]. 43 new building blocks have been integrated in the new release, representing a total number of 60 blocks (from 17 included in the first release), adding interoperability to 6 different biomolecular tools. Work towards the new proposed exascale biomolecular workflows in BioExcel-2 required new useful tools to be wrapped into the library. The novel identified building blocks were presented in the D2.1 document, together with an initial roadmap for the first year of the BioExcel-2 period. During this first year, three new categories have been implemented, with a clear focus on drug discovery related projects, fundamental for the use cases proposed in the new BioExcel-2 period: chemistry, structure_utils and pmx. In particular, the new modules will be all used in the *rational drug design* use case workflows, where functionalities such as PDB information extraction, automatic ligand parameterization, and free energy calculations are needed.

2.2.1 Chemistry module

Cheminformatics packages are essential tools in drug discovery pipelines. They offer the possibility to search, convert, analyze and perform other operations with chemistry data such as small molecules. **OpenBabel** [10] (The Open Source Chemistry Toolbox) is a popular open source software able to read, write and convert over 110 different chemical file formats. **ACPype** [11] is a useful tool able to produce topology files based on the Generalized AMBER force-field (GAFF [12]) for the main MD software packages available: GROMACS, AMBER, CHARMM and CNS/XPLOR. **Ambertools** [13] is a suite of MD setup and analysis tools offered free of charge by the AMBER team. In particular, a couple of programs contained in the suite are of particular importance for the chemistry of small compounds: **Reduce** and **Antechamber**. **Reduce** is used for properly adding hydrogen atoms to a PDB molecular structure file, following a set of predefined rules. **Antechamber** is designed to create force field parameters for general organic molecules, and is used by **ACPype**. Using these tools, building blocks essential for drug discovery processes involving computational simulations have been defined: generation of small molecules topologies, format conversions, addition and removing of hydrogen atoms, and energy minimization. All these new building blocks were used together in a ligand parameterization demonstration workflow, presented in section 3.1 (initial roadmap), objective B, and available through the biobb tutorials website. The current set of building blocks produced is listed in the Table 1.

Building block	Wrapped tool	Description
AcpypeParamsAC	ACPype	Small molecule parameterization and topology file generation for AMBER MD package.
AcpypeParamsCNS	ACPype	Small molecule parameterization and topology file generation for CNS/XPLOR MD package.
AcpypeParamsGMX	ACPype	Small molecule parameterization and topology file generation for GROMACS MD package.
AcpypeParamsGMXOPLS	ACPype	Small molecule parameterization and topology file generation for OPLS/AA MD package.
BabelConvert	OpenBabel	Small molecule format conversion.
BabelAddHydrogens	OpenBabel	Adds hydrogen atoms to small molecules.
BabelRemoveHydrogens	OpenBabel	Removes hydrogen atoms from small molecules.
BabelMinimize	OpenBabel	Energetically minimize small molecules.
ReduceAddHydrogens	AmberTools Reduce	Adds hydrogen atoms to small molecules.
ReduceRemoveHydrogens	AmberTools Reduce	Removes hydrogen atoms from small molecules.

Table 1.- List of biobb_chemistry building blocks.

2.2.2 Structure utils module

Although there is a long-standing debate about the limitations of the PDB format to be used in computational biomolecular simulations, there is no doubt that it is still the central file format to be considered. Most of the programs working with macromolecular 3D-structures use and/or produce PDB files. In particular, the most popular MD packages use PDB files as a starting structure.

Very recently (April 2019), the PDB consortium, managed by the Worldwide Protein Data Bank (wwPDB), an international partnership that collaboratively oversees deposition, validation, biocuration and open-access dissemination of 3D macromolecular structure data, announced that as of 1 July 2019, PDBx/mmCIF will be the only format allowed for deposition of the atomic coordinates for PDB structures resulting from macromolecular crystallography, including X-ray, neutron, fiber and electron diffraction methods [14]. This change triggered a mandatory adaptation for all the tools now working with PDB files to be compatible with the new format, a change that is slowly and progressively occurring in the biomolecular simulation field.

One of the old PDB format limitations is the difficulty to parse and extract information contained within, which is actually harder when working with bigger molecules such as macromolecular protein complexes usually including many different chains, ligands, and metal ions. With that in mind, a collection of utilities to work with (i.e. extract, remove, renumber) this useful information from PDB files has been developed and wrapped into our BioExcel building blocks. The set of building blocks produced for this category is listed in the Table 2. The updated roadmap presented in this document will highlight the extension of this category with building blocks to work with the new PDBx/mmCIF format.

Building block	**Wrapped tool**	**Description**
CatPDB	in house	Class to concatenate two PDB structures in a single PDB file.
ExtractAtoms	in house	Class to extract a selection of atoms from a 3D structure.
ExtractChain	in house	Class to extract a chain from a 3D structure.
ExtractHeteroAtoms	in house	Class to extract hetero-atoms from a 3D structure.
ExtractModel	in house	Class to extract a model from a 3D structure.
ExtractProtein	in house	Class to extract a protein from a 3D structure.
RemoveLigand	in house	Class to remove the selected ligand atoms from a 3D structure.
RemovePdbWater	in house	Class to remove water molecules from PDB 3D structures.
RenumberStructure	in house	Class to renumber atomic indexes from a 3D structure.
SortGroResidues	in house	Class to sort the selected residues from a GRO 3D structure.

Table 2.- List of biobb_structure_utils building blocks.

2.2.3 pmx module

pmx [15] is one of the key tools for the BioExcel Centre of Excellence. Free energy calculations are extremely popular nowadays in the biomolecular computation field, especially in drug discovery. pmx is also a central tool for two of the use cases being studied in the second period of the project: antibody design and rational drug design. The first pmx functionalities implemented in the library allows topology generation for amino acid mutations. The new building blocks were used together in a protein mutation free energy calculation demonstration workflow, presented in section 3.1 (initial roadmap), objective A, and available through the biobb tutorials website. They are also being used in the first workflow of the *rational drug design* use case, studying the impact of amino acid mutations to ligand binding affinities in a particular protein system (EGFR). The three main tools for protein amino acid mutations inside the pmx package were wrapped and offered as BioExcel building blocks, and are listed in Table 3.

Building block	Wrapped tool	Description
Mutate	pmx	pmx tool to insert mutated residues in structure files for free energy simulations
Gentop	pmx	pmx tool to generate hybrid GROMACS topologies: adding a B state to an .itp or .top file for a hybrid residue
Analyse	pmx	pmx tool to calculate free energies from fast growth thermodynamic integration simulations.

Table 3.- List of biobb_pmx building blocks.

2.3 Updated categories

The BioExcel building blocks library is in continuous development. The iterative software development process allows agile updates on the code, even between main releases. During the first year of the BioExcel-2 project, small updates have been performed to some of the biobb categories (biobb_io, biobb_md), whereas other categories have seen a substantial change (biobb_analysis), with a large increase in the number of building blocks included. Updated categories and building blocks since the last release of the library (March 2019) are described in the next sections.

2.3.1 IO module

The biobb_io module is a clear example of continuous development. Being a category defined as input/output of biological information, it basically relies on available biological databases, typically providing REST API services to retrieve data from. The first building blocks developed within the module were able to download PDB structures, map a list of mutations (variants) mapped to a PDB code, and list PDB files of a cluster with a certain sequence similarity to a given structure. The list of building blocks could (and will) be extended with queries to many different databases. During the period reported, a new building block has been included, the *Ligand* bb, which retrieves a PDB file with atomistic information for a particular ligand, given a 3-letter identifier. The updated list of building blocks available from the biobb_io module is listed in the Table 4. A roadmap for the planned i/o building blocks to be developed can be found in the *Updated Roadmap* section.

Building block	Wrapped tool	Description
Ligand	API Call	Downloads a ligand file from the MMB REST API.
Pdb	API Call	Downloads a PDB file from the RCSB or MMB REST APIs.
MmbPdbVariants	API Call	Creates a text file containing a list of all the variants mapped to a RSCB PDB code from the corresponding UNIPROT entries.
MmbPdbClusterZip	API Call	Creates a zip file containing all the PDB files in the given sequence similarity cluster percentage of the given RSCB PDB code.

Table 4.- List of biobb_io building blocks.

2.3.2 MD module

The biobb_md is one of the central modules of the BioExcel building blocks library. MD simulations are central to the project, and thus, this category is another example of continuous development. In this period of the project, features to add free energy calculations have been developed. Changes integrated involved a new pre-defined GROMACS configuration file (free energy, see biobb_md grompp documentation) in the biobb_md grompp building block, and the addition of a new output file (dhdl, see biobb_md mdrun documentation) in the biobb_md mdrun building block. In this case, the list of building blocks available from the biobb_md module has not changed since the last release, but for the sake of completeness of the document, the available list is also presented in the Table 5. A roadmap for the planned md building blocks to be developed can be found in the *Updated Roadmap* section.

Building block	Wrapped tool	Description
Pdb2gmx	gmx pdb2gmx	Creates a compressed (ZIP) GROMACS topology (TOP and ITP files) from a given PDB file.

Building block	Wrapped tool	Description
Editconf	gmx editconf	Creates a GROMACS structure file (GRO) adding the information of the solvent box to the input structure file.
Genion	gmx genion	Creates a new compressed GROMACS topology adding ions until reaching the desired concentration to the input compressed GROMACS topology.
Genrestr	gmx genrestr	Creates a new GROMACS compressed topology applying the indicated force restrains to the given input compressed topology.
Grompp	gmx grompp	Creates a GROMACS portable binary run input file (TPR) applying the desired properties from the input compressed GROMACS topology.
Mdrun	gmx mdrun	Performs molecular dynamics simulations from an input GROMACS TPR file.
MakeNdx	gmx make_ndx	Creates a GROMACS index file (NDX) from an input selection and an input GROMACS structure file.
Solvate	gmx solvate	Creates a new compressed GROMACS topology file adding solvent molecules to a given input compressed GROMACS topology file.
Ndx2resttop	in house	Creates a new GROMACS compressed topology applying the force restrains to the input groups in the input index file to the given input compressed topology.

Table 5.- List of biobb_md building blocks.

2.3.3 Analysis module

The biobb_analysis module has been substantially modified since the last release. A large set of analysis tools has been added, including tools wrapped from the analysis set available in the GROMACS package and building blocks built using the cpptraj [16] program of Ambertools. The use of cpptraj gives the module more flexibility in the trajectory formats to be used and produced by the analysis building blocks. The updated list of building blocks available from the biobb_analysis module is listed in the Table 6. A roadmap for the planned analysis building blocks to be developed can be found in the *Updated Roadmap* section.

Building block	Wrapped tool	Description
GMXCluster	gmx cluster	Creates cluster structures from a given GROMACS compatible trajectory.

Building block	Wrapped tool	Description
GMXRms	gmx rms	Performs a Root Mean Square deviation (RMSd) analysis from a given GROMACS compatible trajectory.
GMXRgyr	gmx gyrate	Computes the radius of gyration (Rgyr) of a molecule about the x-, y- and z-axes, as a function of time, from a given GROMACS compatible trajectory.
GMXEnergy	gmx energy	Extracts energy components from a given GROMACS energy file.
GMXImage	gmx trjconv	Corrects periodicity (image) from a given GROMACS compatible trajectory file.
GMXTrjconvStr	gmx trjconv	Converts between GROMACS compatible structure file formats and/or extracts a selection of atoms.
GMXTrjconvStrEns	gmx trjconv	Extracts an ensemble of frames containing a selection of atoms from GROMACS compatible trajectory files.
GMXTrjconvTrj	gmx trjconv	Converts between GROMACS compatible trajectory file formats and/or extracts a selection of atoms.
Average	Ambertools cpptraj	Calculates a structure average on a given cpptraj compatible trajectory.
Bfactor	Ambertools cpptraj	Calculates the B-factor fluctuations on a given cpptraj compatible trajectory.
Rms	Ambertools cpptraj	Calculates the Root Mean Square deviation (RMSd) on a given cpptraj compatible trajectory.
Rmsf	Ambertools cpptraj	Calculates the Root Mean Square fluctuations (RMSf) on a given cpptraj compatible trajectory.
Rgyr	Ambertools cpptraj	Computes the radius of gyration (Rgyr) from a given cpptraj compatible trajectory.
Dry	Ambertools cpptraj	Dehydrates a given cpptraj compatible trajectory stripping out solvent molecules and ions.
Strip	Ambertools cpptraj	Strips a defined set of atoms (mask) from a given cpptraj compatible trajectory.

Building block	Wrapped tool	Description
Snapshot	Ambertools cpptraj	Extracts a particular snapshot from a given cpptraj compatible trajectory.
Slice	Ambertools cpptraj	Extracts a particular trajectory slice from a given cpptraj compatible trajectory.
Convert	Ambertools cpptraj	Converts between cpptraj compatible trajectory file formats and/or extracts a selection of atoms or frames.
Mask	Ambertools cpptraj	Extracts a selection of atoms from a given cpptraj compatible trajectory.
Image	Ambertools cpptraj	Corrects periodicity (image) from a given cpptraj trajectory file.

Table 6.- List of biobb_analysis building blocks.

2.4 New functionalities

Common functionalities used by the whole collection of building blocks are detached from the specific categories and placed in an additional module (biobb_common). Examples of these functionalities include command-line execution of the wrapped tool, managing of log files, creation and removing of temporary folders required for execution, zipping/unzipping compressed files, unit testing auxiliary functions, etc. The *common* module is also in continuous development, and the new functionalities added in the current release are presented in the following sections.

2.4.1 Container execution compatibility

The software development best practices followed in the packaging of the BioExcel building blocks library make them easy to install thanks to the Conda environments and container infrastructures. Sometimes, though, software incompatibilities hinder the building of a single Conda package with all the required dependencies inside. An example of this case occurred when trying to build the pmx software wrappers. As pmx is coded in Python2 and our building blocks library is using Python3, the automatic packaging process was failing. Actually, combining executions of Python v2 with Python v3 programs in the same computer is not a straightforward process, typically requiring a careful environment configuration, one for each version, especially when they need version-specific libraries. This particular problem will be fixed during the lifespan of BioExcel-2, as pmx is currently being ported to the Python3 version, but to avoid a similar problem with different software tools in the future, a new functionality was included in the *biobb* common module: the container execution compatibility.

In the current release, the library building blocks are able to wrap not just program binaries, but also Docker and Singularity container executions. Software incompatibilities can be avoided using the container technology, where an application is packaged so it can be run, with its dependencies, isolated from other processes. The example used to test the new development was the pmx package. The Python2 version of pmx was encapsulated together with all its dependencies in a Docker container (available in the BioExcel GitHub repository). Wrapping this Docker container instead of the Python2 version of the pmx tool, the building of a Conda package for the biobb_pmx library module was possible.

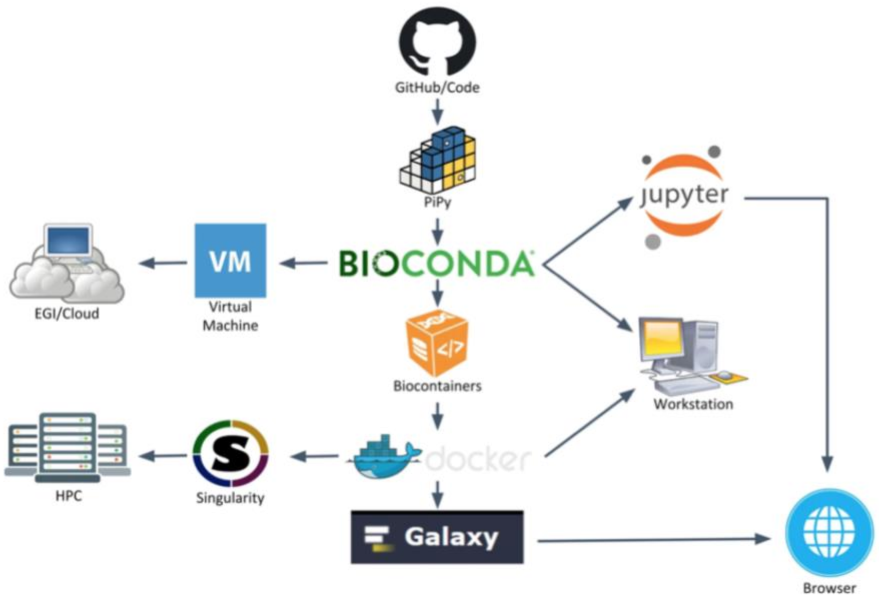


Figure 2.- Distribution and deployment flow for the BioExcel building blocks library. Image taken from reference [7]

This approximation provides more flexibility to the library wrappers at the cost of adding an extra layer of complexity. The distribution and deployment flow (Fig. 2) followed during the development of each library module contains a step where the BioConda package is converted to a (bio)container. Converting these building blocks to Docker/Singularity containers will require being able to launch containers inside a container, which is possible, but considered a not recommended practice. Besides, it will require a manual configuration of the Docker containers generated from the building blocks, as the automatic build from BioConda is not compatible with this approach. All these points will be explored in the next steps of the development, balancing between pros and cons of the methodology.

2.4.2 Workflow parameters

The BioExcel building blocks library has been designed with a main focus: building complex biomolecular simulation workflows. The interoperability offered by the wrapping philosophy allows easy interconnection between the different tools, whereas the adapter layer (biobb adapters) makes them compatible with a number of workflow managers. Accordingly, a new set of parameters restricted to workflow execution has been added to the common building blocks functionality module (biobb_common).

The *working_dir_path* parameter allows imposing a name to a new folder where all the workflow steps outputs will be generated. The *can_write_console_log* and the *remove_tmp* parameters are useful for debugging purposes. The first one redirects the output logs to the terminal console, whereas the second one avoids removing the temporary files generated by every step of the workflow. Finally, the *restart* parameter checks for already generated outputs, avoiding re-executions of workflow steps, useful in complex workflows involving a large number of steps. All definitions can be found in the configuration settings module from the biobb_common API documentation.

2.4.3 Building Blocks common parameters

Analogously to the workflow parameters, the individual building blocks also share a collection of common parameters defined on top of the ones specifically related to the tool wrapped. In the current version, these parameters are basically used to overwrite the behavior of the previously described workflow directives. For example, if there is an interest in re-running a particular workflow by just changing an input configuration file for a single step of the pipeline, one should explicitly define the *restart* parameter for this particular step (and for the ones using its new output) to *false*, which will force the execution of the step(s) with the new configuration, whereas for the rest of the steps, outputs from previous executions will be used. The definitions for these parameters can also be found in the configuration settings module from the biobb_common API documentation.

Besides these properties to overwrite global parameters, every building block from the library now has a collection of specific parameters related to the container execution (see section 2.4.1). These parameters are:

- **container_path**: Path to the binary executable of the container.
- **container_image**: Container Image identifier.
- **container_volume_path**: Path to an internal directory in the container.
- **container_working_dir**: Path to the internal CWD in the container.
- **container_user_id**: User number id to be mapped inside the container.
- **container_shell_path**: Path to the binary executable of the container shell.

The combination of these properties offers a great flexibility on the container image to use, on the tool being wrapped, and on the parameters used inside the container, such as temporary folders to generate the output files, user id to execute the binaries, and even the shell to be used, which can depend on the base system included in the container.

2.5 Bug fixes

A practice present in every software development process is the identification and fixing of errors or software bugs. Small, non-critical errors affecting a specific biobb category are fixed and briefly documented in the issues section of the corresponding biobb GitHub repository. The issues section of the central biobb GitHub repository (biobb) is used to collect and document general issues for the whole library. One important bug identified and fixed since the last release has been selected to illustrate the work done in this part of the development process, and is briefly explained in the following paragraphs.

Every building block implemented in the library automatically generates a couple of log files during its execution runtime: a standard output log and a standard error log, gathering the output information of the tool wrapped. During the library testing process using complex workflows, an issue related to these log files was identified. The particular workflow being tested contained a large loop running a sub-workflow defined by 11 different building blocks. The combination of a large number of building blocks in a single workflow execution raised a "*too many files open"* Linux error. After debugging of the code, a bug involving the creation of the output and error log files was discovered. Some of the file descriptors were not correctly closed, with the result being a number of accumulated open files that made the workflow crash when reaching the operative system limit. This bug was complex to find and fix, as it required a large number of iterations in a single workflow execution to be reproduced.

The bug was solved introducing a Python decorator, a Python object used to modify a function or a class. The function modified was the common *launch* function, which contains the system call to the tools being wrapped. This function is used by all the building blocks, and is the one that generated the log files. The new decorator (*@launchlogger, file_utils.py*) is now dealing with the opening and closing of the log files, solving the problem, and at the same time cleaning the code in the *launch* function.

Several other bugs were identified and fixed since the last software release, most of them related to missing documentation or common functionalities included in the biobb_common module. Progress of bugs identified from GitHub issues can be tracked in the GitHub issues web-based interface of each particular biobb module, whereas bugs identified internally can be tracked using the commits history of the same repository. As an example, the module with the highest number of issues is the biobb_md, and can be tracked from https://github.com/bioexcel/biobb_md/issues, while bugs fixed from the common biobb_common module can be tracked from the commits history https://github.com/bioexcel/biobb_common/commits/master.

2.6 Visibility and outreach

The BioExcel building blocks library has reached a good level of maturity. A publication by Andrio et al. [7] describes the library together with the best practices followed during its development process. Complex scientific workflows were built and tested in different infrastructures: local desktops, public and private cloud infrastructures (VMs), web servers, and HPC supercomputers (CPU and GPU-based). A pre-exascale massive execution was run in Marenostrum supercomputer using 40,000 cores as a showcase of the library power. The last release includes not only new building blocks, but also new features. With this point reached, one of the main goals from now on is to increase the visibility and outreach of the library. The first steps towards this goal are presented in the next sections.

2.6.1 Landing website

The biobb library is formed by a large collection of different modules, corresponding to tools wrappers from various categories: chemistry, md, analysis, etc. Every module is made available in different ways: source code (GitHub), packages (BioConda), containers (Docker, Singularity). Every module has its own documentation pages (readthedocs). There are multiple ways of installing and executing the library (Jupyter notebooks, Galaxy, command-line). All together, the library is a compendium of distributed pieces, which required a central landing point gathering all the information together, listing links to all the external resources, and offering tutorials on how to find, install, and use the library. This central infrastructure is a new webpage accessible via http://mmb.irbbarcelona.org/biobb/ (Fig. 3).

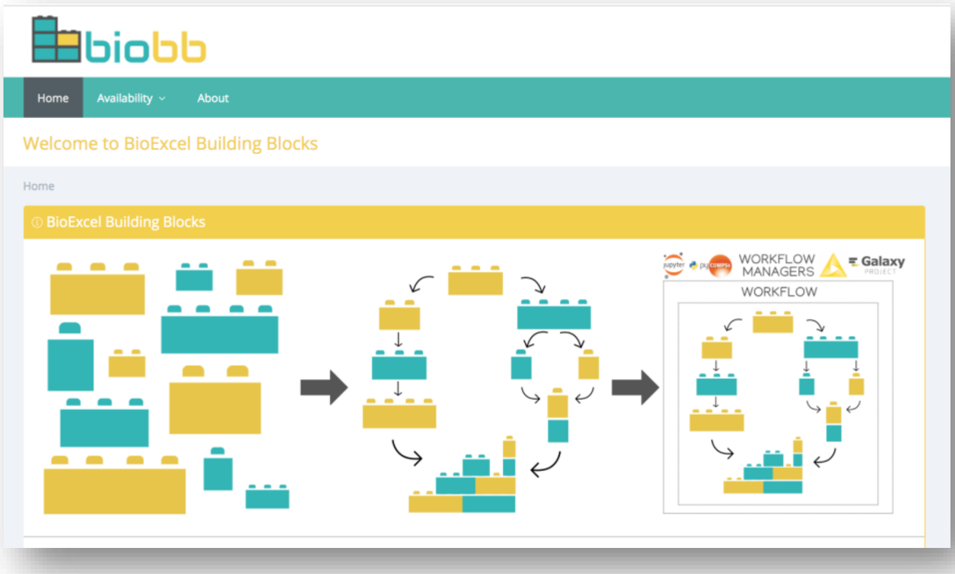


Figure 3.- BioExcel building blocks (biobb) landing website: http://mmb.irbbarcelona.org/biobb/

The website has 4 main sections under the *Availability* tab: *Launch* and *Download & install* present the different ways available to download, install and run the library, with the corresponding links to resources. *Source and Docs* presents a list of the complete set of building blocks divided in the different available categories. A convenient particular feature of this section is that the list is automatically retrieved from the GitHub repositories, so any time a new commit arrives, an update of the list is triggered (GitHub webhooks), ensuring coherence between what is available and what is presented. Finally, the *Tutorials* section is designed to be a live, extensible page, where Jupyter Notebook tutorials showing demonstration workflows and properly documented examples will be hosted.

2.6.2 Jupyter Notebooks tutorials

Jupyter Notebooks, extensively described in the last WP2 BioExcel-2 deliverable (D2.1), are an extension of the iPython (interactive Python) initiative, a command shell for interactive computing in multiple programming languages (initially developed for Python). Jupyter Notebooks extend the console-based interactive shell providing a web-based application with which programmers can develop, document, execute code, and share results. This web-based application is ultimately acting as a Graphical User Interface (GUI) that allows the graphical representation and visualization of charts, tables, molecular visualizers, and in general any kind of integration offered by the compatible libraries. This graphical interface makes Jupyter Notebooks perfect for educational purposes, in the form of tutorials made of interactive programming code accompanied by text information and/or documentation, versatile graphical charts and data visualization, which can be easily exported to printed material.

With that in mind, a new section was created in the biobb website, with a set of Jupyter notebook tutorials showing the possibilities of the library and teaching how to build these examples from scratch. All the tutorials hosted will have a link to the corresponding GitHub repository, readthedocs and web-based documentation, and the possibility to be directly run in the myBinder platform. The page is thought to be dynamic, with new examples being added when implemented. It will also contain tutorials on how to build and run workflows using the biobb library in different workflow managers and infrastructures. The first set of already available tutorials includes 4 illustrative workflows made of BioExcel building blocks (Fig. 4):

- **Protein MD Setup**: Step-by-step tutorial illustrating the process of setting up a simulation system containing a protein, based on the Justin A. Lemkul GROMACS tutorial. Lysozyme (PDB code 1AKI) is used as an example. biobb modules used: io, model, md and analysis.

- **Automatic Ligand Parameterization**: Step-by-step tutorial illustrating the process of ligand parameterization for a small molecule. Ibuprofen (3-letter code IBP, Drugbank code DB01050), a non-steroidal anti-inflammatory drug (NSAID) is used as an example. biobb modules used: io and chemistry.

- **Protein-complex MD Setup**: Step-by-step tutorial illustrating the process of setting up a simulation system containing a protein in complex with a ligand. T4 lysozyme with the double mutation L99A/M102Q (PDB code 3HTB), in complex with the 2-propylphenol (3-letter code JZ4) is used as an example. biobb modules used: io, model, structure_utils, chemistry, md and analysis.

- **Mutation Free Energy Calculations**: Step-by-step tutorial illustrating how to compute a fast-growth mutation free energy calculation, based on the official pmx tutorial. Staphylococcal nuclease (PDB code 1STN), a small, minimal protein, is used as an example being appropriate for a short tutorial. biobb modules used: pmx, md and analysis.

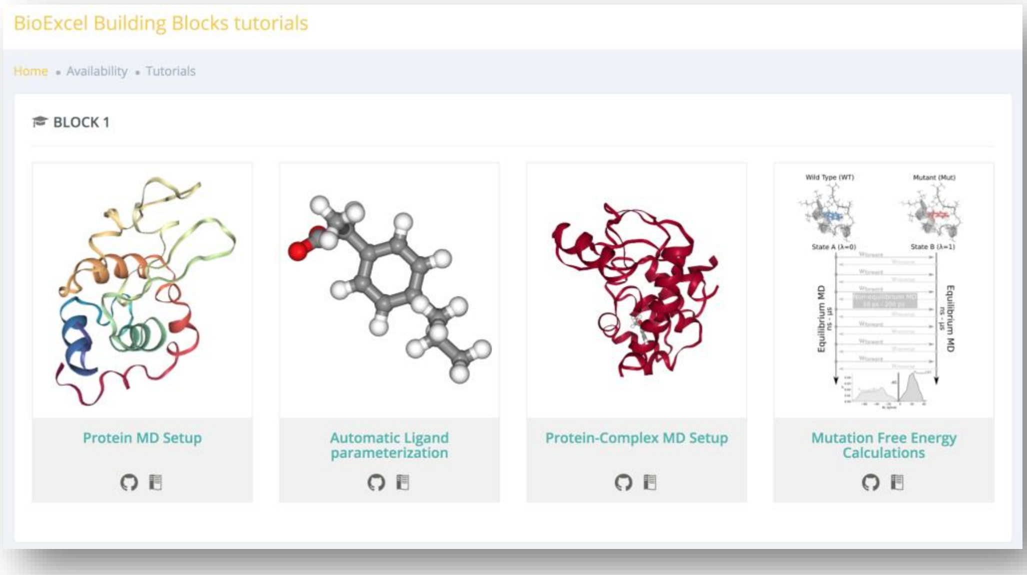

BioExcel Building Blocks tutorials
Home • Availability • Tutorials
BLOCK 1
Protein MD Setup
Automatic Ligand parameterization
Protein-Complex MD Setup
Mutation Free Energy Calculations

Figure 4.- BioExcel building blocks (biobb) website tutorial section. ***http://mmb.irbbarcelona.org/biobb/availability/tutorials/***

These tutorials will become a key part of the development process, as they will be used to:

1) Advertise the library and show its power and possibilities.
2) Teach how to use the library with different workflow managers and infrastructures.
3) Run training events, using the Jupyter Notebooks and the BioExcel Cloud Portal.

It should be noted here that a possible local implementation of the myBinder platform (BinderHub) is currently being explored for deployment in the BioExcel Cloud Portal (see updated roadmap). This will allow us to link our Jupyter Notebook tutorials to a private, controlled infrastructure, which will overcome possible overload problems of the public myBinder server, and will allow us to run training events using this internal interface, user-restricted through BioExcel credentials.

3 Initial & Updated Roadmap

The first WP2 deliverable in the BioExcel-2 period presented an initial roadmap for the first year of the BioExcel building blocks library. The roadmap contained explicit objectives and milestones that are being revised in the next sections. An updated roadmap for the next 2 years till the end of the project has been prepared, and is presented here.

3.1 Initial Roadmap

The initial roadmap for the BioExcel building blocks library, corresponding to Task *T2.1: Application building blocks for computational biomolecular simulations,* contained 3 main objectives, with 3 milestones each:

(A) Building Blocks for the main codes: HADDOCK, pmx, and CP2K

One of the main objectives for BioExcel-2 is to build workflows integrating the functionality and power of the main biomolecular simulation codes in BioExcel: GROMACS, HADDOCK, pmx, and CP2K. After generating a set of wrappers for GROMACS package tools in BioExcel-1 (biobb_md), the first period of BioExcel-2 has been focused on developing building blocks wrapping the pmx functionalities (biobb_pmx). The possibility of wrapping the HADDOCK software has also been explored. We already have Docker and Singularity containers available, which will be used from a new set of building blocks, thanks to the new added container compatibility.

Milestones reached:

- **(A1) Implement building blocks for pmx**:
  Building blocks wrapping pmx functionalities for protein amino acid mutations are ready and available from the corresponding GitHub repository. BioConda package and readthedocs documentation are also available from the same repository. Docker and Singularity containers for this category are still being explored, due to the need of running a container inside a container (see section 2.4.1).

- **(A2) Build a free-energy workflow integrating GROMACS & pmx** and **(A3) Make the workflow available through the GitHub repository and the BioExcel Cloud Portal**:
  Demonstration tutorial workflow ready and available from the biobb tutorials website and from its corresponding GitHub repository. Scientific workflow used in the use case 3 (Rational Drug Design) ready and available from the GitHub repository. Availability of the tutorial from the BioExcel Cloud Portal is being explored (see section 2.6.2).

(B) Building Blocks for the development of the Rational Drug Design use case

One of the proposed use cases for BioExcel-2, the Rational Drug Design use case, addresses studies of interest to the pharmaceutical industry. It is an ambitious project divided into four inter-dependent workflows, presented in order of increasing complexity. New biobb categories will involve chemistry tools, biased MD methods (REMD, TMD, and SMD), or MD setup for membrane-embedded proteins. The first period of BioExcel-2 has been focused on developing building blocks wrapping cheminformatics tools (biobb_chemistry), performing useful tasks such as small molecules format conversion, ligand parameterization or 3D structure generation.

Milestones reached:

- **(B1) Implement building blocks for chemistry tools**:
  Building blocks wrapping cheminformatics functionalities are ready and available from the corresponding GitHub repository. BioConda package, Docker and Singularity containers and readthedocs documentation are also available from the same repository.

- **(B2) Build a ligand parameterization workflow using the new biobb category** and **(B3) Make the workflow available through the GitHub repository and the BioExcel Cloud Portal**:
  Demonstration tutorial workflow ready and available from the biobb tutorials website and from its corresponding GitHub repository. Availability of the tutorial from the BioExcel Cloud Portal is being explored (see section 2.6.2).

(C) Building Blocks for the development of High Performance Data Analytics (HPDA)

HPDA, the use of High Performance Computing (HPC) to analyze large datasets for patterns and insights, is one of the main areas for the BioExcel-2's WP2. The use of HPC parallel processing to run powerful data analysis software tools opens the possibility to examine large datasets within a reasonable time. Work has started using the Distributed Computing Library (dislib), integrated in the PyCOMPSs framework. A dimensionality reduction functionality, popular in the molecular dynamics field, has been implemented in the dislib library: Principal Component Analysis (PCA). The addition of a clustering method [17], also widely used in the molecular dynamics field, is being explored. Building blocks for these functionalities, and other interesting Machine Learning methods, are planned, but not yet implemented.

Milestones **still not** reached:

- **(C1) Implement building blocks for HPDA tools, wrapping dislib utilities:**
  New useful functionalities identified and being developed in the dislib library, building blocks wrapping them still to be developed.

- **(C2) Build an HPDA workflow prototype using the new biobb category** and **(C3) Make the workflow available through the GitHub repository and the BioExcel Cloud Portal**:
  Development of demonstration workflows needs to wait for the new building blocks to be available.

3.2 Updated Roadmap

The Task 2.1, *Application building blocks for computational biomolecular simulations*, has already gained momentum. Three new modules for the BioExcel building blocks have been built (chemistry, pmx, structure_utils), allowing more complex workflows to be assembled. The development process designed in the first period of BioExcel and published in a journal paper is now highly automated and stable. Although the task was officially determined to be active for the first 18 months of the project, continuous updates of the library and development of new building blocks will be needed to help with the complex workflows proposed in the use cases, in particular for the *rational drug design* study. Software development process for the BioExcel building blocks library will keep up the pace gained and continue with its update and extension till the end of the BioExcel-2 period. The focus for the next few years will be put on: new functionalities still missing in the building blocks collection, such as HPDA, biased molecular dynamics or classical molecular interaction potential calculations (i.e. electrostatic, solvation energies); building blocks for the rest of the main codes (HADDOCK, CP2K); and work on maintaining and updating existing modules. An updated roadmap for the PM12-36 follows:

T2.1 - Application building blocks for computational biomolecular simulations

Summary

- Task leader: IRB
- Task partners: BSC, KTH, UU, UNIMAN, EMBL-EBI
- Task project months: 1-18

Technology: Python, (Bio)Conda, Docker, Singularity, readthedocs, CWL, openAPI.

Milestones: Results will be presented in D2.7: Final Release of demonstration workflows (PM36)

(A) Building Blocks for the main codes: HADDOCK, pmx, and CP2K (cont)

Objective A was extensively presented in the D2.1, seeking for the integration of the main BioExcel biomolecular simulation codes GROMACS, HADDOCK, pmx, and CP2K in the BioExcel building blocks library. Milestones A1, A2 and A3 were reached (see previous section). Future work will be focused mainly on HADDOCK and CP2K. The new milestones for this objective are:

New Milestones:

- (A4) Implement building blocks for HADDOCK, starting by HADDOCK2 version packaged in a Docker/Singularity container, followed by the now in-development HADDOCK3 modular source code (see D1.3); build a protein-protein or protein-peptide docking workflow using them, and make it available through the GitHub repository and the BioExcel Cloud Portal.

- (A5) Implement building blocks for CP2K, build a QM energy refinement workflow using them, and make it available through the GitHub repository and the BioExcel Cloud Portal.

(B) Building Blocks for the development of the Rational Drug Design use case (cont)

Objective B was extensively presented in D2.1, seeking for the integration of tools related to drug design processes in the BioExcel building blocks library.

These tools will involve cheminformatics programs, docking methodologies, biased MD methods (REMD, TMD, and SMD), or MD setup for membrane-embedded proteins. Milestones B1, B2 and B3 were reached (see previous section). Work now will be focused mainly on updating and extending the chemistry, virtual screening and md biobb modules with the new drug design-related functionalities. The new milestones for this objective are:

New Milestones:

- (B4) Update the biobb_vs module with new building blocks for Virtual Screening (VS), build a protein-ligand docking workflow using them, and make it available through the GitHub repository and the BioExcel Cloud Portal.
- (B5) Implement building blocks to compute Classical Molecular Interactions Potentials (CMIP), build a demonstration workflow using them, and make it available through the GitHub repository and the BioExcel Cloud Portal.
- (B6) Update the biobb_chemistry module with new building blocks wrapping the RDKit package, build a demonstration workflow using them, and make it available through the GitHub repository and the BioExcel Cloud Portal.
- (B7) Update the biobb_md module including biased and membrane-embedded Molecular Dynamics calculations. Implement building blocks wrapping the PLUMED tool functionalities, build a demonstration workflow using them, and make it available through the GitHub repository and the BioExcel Cloud Portal.

(C) Building Blocks for the development of High Performance Data Analytics (HPDA) (cont)

Objective C was extensively presented in D2.1, seeking for the integration of tools related to HPDA in the BioExcel building blocks library. These tools will involve Machine Learning and the use of High Performance Computing (HPC) to analyze large datasets for patterns and insights. Milestones C1, C2 and C3, the generation of wrappers and workflows for the dislib library, have not been reached yet (see previous section). Work will continue to reach these milestones, and new work will be started in the development of wrappers on top of popular Machine Learning tools. The new milestones for this objective are:

New Milestones:

- (C4) Implement building blocks wrapping Machine Learning (ML) and Deep Learning (DL) tools such as TensorFlow, scikit-learn, pyTorch and Keras, build demonstration workflows using them, and make them available through the GitHub repository and the BioExcel Cloud Portal.

(D) Maintain, update and extend biobb modules:

Objective D is not defining any new biobb category, but is about the process of maintaining, updating and extending the already existing ones. New useful analyses on top of the trajectories generated from MD simulations have been identified during the generation of the first scientific workflows in BioExcel-2. Programs such as MDAnalysis [18] or Bio3D [19] offer a broad collection of analyses that can be integrated in the library. Easy retrieval of information from different biological databases will be very beneficial in our biomolecular workflows, especially using the unique, interoperable syntax offered by the building blocks library. New information available such as the one accessible from the PDBe-KB [20] database is of great interest for the structural community. New functionalities added to the pmx package such as the ones related to nucleic acids and ligand modifications will be also incorporated in the biobb_pmx module.

New Milestones:

- (D1) Update the biobb_analysis module with new building blocks wrapping the MDAnalysis and the Bio3D tools, build demonstration workflows using them, and make them available through the GitHub repository and the BioExcel Cloud Portal.

- (D2) Update the biobb_io module with new building blocks retrieving information from useful biological databases such as the PDBe API, build demonstration workflows using them, and make them available through the GitHub repository and the BioExcel Cloud Portal.

- (D3) Update the biobb_pmx module adding new building blocks to support nucleic acids and ligand modifications, build demonstration workflows using them, and make them available through the GitHub repository and the BioExcel Cloud Portal.

4 Conclusions

The first release of the BioExcel building blocks library for the BioExcel-2 period is available: v2.0.0, 2019.4 release. This version extends the last release delivered in the BioExcel-1 period and presented in the published paper *BioExcel Building Blocks, a software library for interoperable biomolecular simulation workflows* [7].

The release is formed by the source code, packages and containers to install and use the library building blocks, and their associated documentation, generated using readthedocs. The collection of building blocks are divided into categories according to the functionality of the tools wrapped. Every category is available through the GitHub repository, BioConda packages and Docker/ Singularity containers, easing the access and usage of the library building blocks.

Software updates and bug fixes have been applied to the library, and three new modules, wrapping PDB structures parsing tools, cheminformatics tools, and the pmx package have been added. These modules will allow the building of complex biomolecular workflows, with particular emphasis on free energy calculations and drug discovery processes, functionalities aligned with the use cases being studied in the project, especially with the *rational drug design* project. A landing website has been designed and implemented, with the goal of being the central, one-stop shop place, where all the information regarding the BioExcel building blocks will be gathered, a step forward to increase the visibility and outreach of the library.

The initial roadmap presented in the D2.1 document has been revised. Work done during the first year of the BioExcel-2 period is in good agreement with the roadmap proposed, with only a small delay on the HPDA building blocks. A new, updated roadmap for the next two years of the project has been planned. The roadmap includes the generation of wrappers for the collection of categories and tools identified and presented in the D2.1 document.

The BioExcel building blocks is now a mature software library, with a development plan consisting of 4 releases per year. The extension of the library with PDB structure utils, pmx and cheminformatics tools has increased the complexity of the workflows assembled using the building blocks. The inclusion of new functionalities planned such as HPDA methodologies will allow the use of HPC parallel processing to run powerful data analysis. A set of demonstration workflows using the new library functionalities, already in development, are going to be presented in the next deliverable *D2.3 – First Release of Demonstration Workflows*, in PM18.

5 References

1. Wilkinson, M.D., et al., *The FAIR Guiding Principles for scientific data management and stewardship.* 2016. **3**: p. 160018. https://doi.org/10.1038/sdata.2016.18
2. Jiménez, R.a.K., M and Alhamdoosh, M and Barker, M and Batut, B and Borg, M and Capella-Gutierrez, S and Chue Hong, N and Cook, M and Corpas, M and Flannery, M and Garcia, L and Gelpí, JL and Gladman, S and Goble, C and González Ferreiro, M and Gonzalez-Beltran, A and Griffin, PC and Grüning, B and Hagberg, J and Holub, P and Hooft, R and Ison, J and Katz, DS and Lesko?ek, B and López Gómez, F and Oliveira, LJ and Mellor, D and Mosbergen, R and Mulder, N and Perez-Riverol, Y and Pergl, R and Pichler, H and Pope, B and Sanz, F and Schneider, MV and Stodden, V and Suchecki, R and Svobodová Vaeková, R and Talvik, HA and Todorov, I and Treloar, A and Tyagi, S and van Gompel, M and Vaughan, D and Via, A and Wang, X and Watson-Haigh, NS and Crouch, S, *Four simple recommendations to encourage best practices in research software* (Jiménez et al. 2017)*[version 1; referees: 3 approved].* F1000Research, 2017. **6**(876). https://doi.org/10.12688/f1000research.11407.1
3. Lamprecht, A.-L., et al., *Towards FAIR principles for research software.* Data Science, 2019. https://doi.org/10.3233/DS-190026
4. Tejedor, E., et al., *PyCOMPSs: Parallel computational workflows in Python.* International Journal of High Performance Computing Applications, 2015. https://doi.org/10.1177/1094342015594678
5. López-Ferrando, V., et al., *PMut: a web-based tool for the annotation of pathological variants on proteins, 2017 update.* Nucleic Acids Research, 2017. **45**(W1): p. W222-W228. https://doi.org/10.1093/nar/gkx313
6. Peter, A., et al., *Common Workflow Language, v1.0.* 2016. https://doi.org/10.6084/m9.figshare.3115156.v2
7. Andrio, P., et al., *BioExcel Building Blocks, a software library for interoperable biomolecular simulation workflows.* Sci Data, 2019. **6**(1): p. 169. https://doi.org/10.1038/s41597-019-0177-4
8. Vivian, J., et al., *Toil enables reproducible, open source, big biomedical data analyses.* Nat Biotech, 2017. **35**(4): p. 314-316. https://doi.org/10.1038/nbt.3772
9. Hospital, A., et al., *BioExcel Deliverable D2.4 - Final Release of Workflows and Modular Tools for User Services.* 2019. https://doi.org/10.5281/zenodo.3236256
10. O'Boyle, N.M., et al., *Open Babel: An open chemical toolbox.* Journal of Cheminformatics, 2011. **3**(1): p. 33. https://doi.org/10.1186/1758-2946-3-33
11. Sousa da Silva, A.W. and W.F. Vranken, *ACPYPE - AnteChamber PYthon Parser interfacE.* BMC Research Notes, 2012. **5**: p. 367-367. https://doi.org/10.1186/1756-0500-5-367
12. Wang, J., et al., *Development and testing of a general Amber force field.* J Comput Chem, 2004. **25**. https://doi.org/10.1002/jcc.20035
13. Case, D.A., et al., *AMBER 2016*, S.F. University of California, Editor. 2016: University of California, San Francisco. https://ambermd.org/doc12/Amber16.pdf

14. Adams, P.D., et al., *Announcing mandatory submission of PDBx/mmCIF format files for crystallographic depositions to the Protein Data Bank (PDB).* Acta Crystallogr D Struct Biol, 2019. **75**(Pt 4): p. 451-454. https://doi.org/10.1107/S2059798319004522
15. Gapsys, V., et al., *pmx: Automated protein structure and topology generation for alchemical perturbations.* Journal of Computational Chemistry, 2015. **36**(5): p. 348-354. https://doi.org/10.1002/jcc.23804
16. Roe, D.R. and T.E. Cheatham, *PTRAJ and CPPTRAJ: Software for Processing and Analysis of Molecular Dynamics Trajectory Data.* J Chem Theory Comput, 2013. **9**(7): p. 3084-95. https://doi.org/10.1021/ct400341p
17. Daura, X., et al., *Peptide Folding: When Simulation Meets Experiment.* Angewandte Chemie International Edition, 1999. **38**(1 ‐ 2): p. 236-240.
18. Michaud-Agrawal, N., et al., *MDAnalysis: A toolkit for the analysis of molecular dynamics simulations.* Journal of Computational Chemistry, 2011. **32**(10): p. 2319-2327. https://doi.org/10.1002/(SICI)1521-3773(19990115)38:1/2<236::AID-ANIE236>3.0.CO;2-M
19. Grant, B.J., et al., *Bio3d: an R package for the comparative analysis of protein structures.* Bioinformatics, 2006. **22**(21): p. 2695-2696. https://doi.org/10.1093/bioinformatics/btl461
20. consortium, P.D.-K., *PDBe-KB: a community-driven resource for structural and functional annotations.* Nucleic Acids Research, 2019. https://doi.org/10.1093/nar/gkz853