

H2020 EINFRA-5-2015



www.bioexcel.eu

Project Number 675728

D2.4 – Final Release of Workflows and Modular Tools for User Services

*WP2: Portable Environments for Computing and
Data Resources*



Copyright© 2015-2018 The partners of the BioExcel Consortium



This work is licensed under a [Creative Commons Attribution 4.0 International License](https://creativecommons.org/licenses/by/4.0/).

The opinions of the authors expressed in this document do not necessarily reflect the official opinion of the BioExcel partners nor of the European Commission.

Document Information

Deliverable Number	D2.4
Deliverable Name	Final Release of Workflows and Modular Tools for User Services
Due Date	2018-10-31 (PM36)
Deliverable Lead	IRB
Authors	Adam Hospital (IRB), Anna Montras (IRB), Josep Lluís Gelpí (BSC), Daniele Lezzi (BSC), Rosa M. Badia (BSC), Steven Newhouse (EMBL-EBI), Jose A. Dianas (EMBL-EBI), Pau Andrio (BSC), Luis Jordà (BSC), Stian Soiland-Reyes (UNIMAN), Darren J. White (UEDIN), Adam Carter (UEDIN), Emiliano Ippoliti (Juelich), Adrien Melquiond (UU), Bert de Groot (MPG)
Keywords	Cloud Computing, HPC, Tools, Workflows
WP	WP2
Nature	Other
Dissemination Level	Public
Final Version Date	201X-XX-XX
Reviewed by	
MGT Board Approval	201X-XX-XX

Document History

Partner	Date	Comments	Version
IRB	2018-09-09	First draft	0.1
IRB, BSC	2018-10-03	First draft reviewed by BSC	0.2
UNIMAN	2018-10-05	Comments/edits by UNIMAN	0.2.1
IRB	2018-10-05	Integration of contributions, comments and edits	0.3
IRB	2018-10-19	Comments from EB review addressed	0.4
IRB, BSC	2018-10-23	FAIR section revised	0.4.1
IRB	2018-10-26	First Final Draft	0.5

Executive Summary

This deliverable presents the final release of workflows and modular tools designed and developed in the BioExcel project during the last three years. A summary of the software development process adopted, following a set of best practices promoted by ELIXIR, is exposed together with examples of working implementations. Availability of the building blocks and workflows produced so far by the project is listed, describing where to find them and how can they be installed and used, presenting a variety of approaches designed for different types of users.

The current stage of the *Model Protein Mutants* transversal workflow, used as a pivot study and demonstrator for the workflows software library development, is revised. Also, two scientific success stories achieved using this pipeline are presented, showing the potential of the methodology in a real scenario. The current implementation for each of the project pilot use cases is detailed, with special focus on the workflows behind them, availability and feedback.

Finally, the services provided and the present phase for the two deployment environments, development and production, is outlined. An overview of the tools developed and those currently available in each of the cloud and HPC infrastructures is given. Particularly successful use cases produced by the BSC testbed and the BioExcel Cloud Portal are depicted.

Contents

1	INTRODUCTION	6
2	TRANSVERSAL WORKFLOW UNIT: <i>MODEL PROTEIN MUTANTS</i>	7
2.1	WORKFLOW OVERVIEW	7
2.2	SOFTWARE LIBRARY	7
2.2.1	PYTHON WRAPPERS	8
2.2.2	WORKFLOW MANAGER-AGNOSTIC PHILOSOPHY	9
2.2.3	INFRASTRUCTURE-AGNOSTIC PHILOSOPHY	11
2.2.4	BEST PRACTICES: FAIR PRINCIPLES	13
2.2.5	COMMON WORKFLOW LANGUAGE	16
2.3	AVAILABILITY	20
2.3.1	EXPERT USERS (COMMAND LINE, HPC)	20
2.3.2	ENTRY-LEVEL USERS (GUI)	22
2.3.3	INTERMEDIATE USERS	23
2.4	TECHNICAL SUCCESS STORY: PYRUVATE KINASE ANNOTATED MUTATIONS	24
2.5	SCIENTIFIC SUCCESS STORY: EPIDERMAL GROWTH FACTOR RECEPTOR CANCER MUTATIONS	26
3	PILOT USE CASES	30
3.1	PILOT USE CASE 1: HIGH THROUGHPUT WORKFLOW FOR CANCER GENOME SEQUENCING DATA	30
3.2	PILOT USE CASE 2: FREE ENERGY SIMULATIONS OF BIOMOLECULAR COMPLEXES	35
3.3	PILOT USE CASE 3: MULTI-SCALE MODELING OF MOLECULAR BASIS FOR ODOR AND TASTE	37
3.4	PILOT USE CASE 4: BIOMOLECULAR RECOGNITION	39
3.5	PILOT USE CASE 5: VIRTUAL SCREENING	41
4	DEPLOYMENT OPTIONS	43
4.1	DEVELOPMENT CLOUD INFRASTRUCTURE (BSC)	43
4.1.1	OVERVIEW	43
4.1.2	AVAILABLE TOOLS	43
4.1.3	USE CASES	44
4.2	PRODUCTION CLOUD INFRASTRUCTURE: BIOEXCEL CLOUD PORTAL (EMBL-EBI)	44
4.2.1	OVERVIEW	44
4.2.2	AVAILABLE TOOLS	46
4.2.3	USE CASES	47
5	CONCLUSIONS & FUTURE WORK	48
6	REFERENCES	50
	APPENDIX	52

1 Introduction

The use of computational workflows has become ubiquitous for data analytics in the field of bioinformatics since the last decade. Workflow Managers like Taverna[1, 2], Galaxy[3], KNIME[4] or Pipeline Pilot give a graphical user interface to assemble and run scientific workflows, being the last two the most commonly used in cheminformatics and computational chemistry[5]. More than [200 workflow systems](#) have been identified, with varying degree of usability, domain-specific bindings and execution models. Repositories like myExperiment[6] allow publishing workflow definitions for further sharing and re-usage. However, no single, universal, modular, software methodology solution to compose such workflows exists. Earlier initiatives like BioCatalogue[7] ([myGrid](#)) and [biomoby](#)[8], tried to define an interoperable ecosystem to run bioinformatics tools, web-services and workflows made out of them. More recently the [Common Workflow Language](#) (CWL) has emerged as a cross-community effort to build a common denominator across workflow systems. Computational infrastructures available to researchers today are also varied, and the choice of adoption depends primarily on the user's expertise and personal network. In this context, [ELIXIR](#), a pan-European infrastructure that coordinates, integrates and sustains bioinformatics resources, is working towards a series of recommendations to organize this ecosystem.

High Performance Computing allows the execution of pipelines in a massively parallel manner, obtaining computational results in days that would have taken years to produce on a single workstation. Computational *cloud infrastructures* and *software containers* provide virtualized environments to package complex software installations and configurations in a portable and reliable way.

With all these key points in mind, WP2, the BioExcel portable environments for computing and data resources work package, designed and presented a roadmap (D2.1) [[10.5281/zenodo.263963](#)], starting with defining a collection of useful software, selected from the partners' expertise, to be used in a bottom-up approach ending up with a set of *biomolecular building blocks* and workflows of particular interest in the field.

BioExcel decided to follow ELIXIR's workflow software development process. While ELIXIR overall can be said to have a bias towards *genomics*, the role of BioExcel within ELIXIR have grown to become the **main contributor of best practices** in the domain of *structural and computational biomolecular science*.

The **BioExcel building blocks**, as first outlined in D2.2 [[10.5281/zenodo.263965](#)], were combined with workflow managers and available computational environments, covering most aspects of biomolecular computational research. Expert, entry-level and intermediate users have access to BioExcel workflow tools, through a broad range of implementations. The final release of workflows and modular tools in BioExcel are presented in this document.

2 Transversal Workflow Unit: *Model Protein Mutants*

2.1 Workflow overview

The transversal workflow “*Model Protein Mutants*” is a pipeline with real biological interest that was proposed in the project Description of Action (Task 2.2) as a transversal unit. This is because it contains building blocks and functionalities that will be also needed in several other studies of interest for the project, as for example remote data access or MD simulation setup and running. It was extensively described in the D2.2 [[10.5281/zenodo.263965](https://doi.org/10.5281/zenodo.263965)], and has been used as a prototype to test not just the workflows development process in the project but also the whole computational infrastructure, being the core of the technical and scientific challenges that are being presented in the following sections.

The *Model Protein Mutants* workflow can be described as an automated protocol to generate and simulate structures for protein variants identified from genomics data. Structures are prepared and analyzed using Molecular Dynamics (MD) simulations. The pipeline receives a *Uniprot* protein id as input, and it automatically retrieves all the information needed to model the structures for the different annotated mutations. It then prepares and runs MD simulations for each of the systems, thus obtaining *dynamic* information (trajectories and ensembles of modeled structures for each of the protein variants), which can be then used in a comparative study. The pipeline uses data from public repositories, but is configurable to accept user-provided specific datasets. For a diagram of the whole pipeline and the processes involved, please refer to the [D2.2, Fig. 4.1](#).

2.2 Software Library

BioExcel WP2 work started with an opening analysis of the state-of-the-art of portable environments for computing. This identified a set of workflow managers, computational infrastructures, data resources, and finally a catalogue of biomolecular computational tools, most of them being supported or used by BioExcel partners. These tools are the main units for the development of our workflows, which have now been converted into our BioExcel building blocks (*biobb*): a set of Python modules (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, putting together ELIXIR’s recommendations and services with biomolecular simulation, thus demonstrating the feasibility of working according to the FAIR principles in this field. FAIR guiding principles[9] of data management puts 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[10], on

the other hand, encourage the adoption of existing best practices in research software development. There is a clear alignment between FAIR principles and OSS recommendations, being the first attempt to define *FAIR principles applied to software*. BioExcel building blocks, thus, is following these FAIR principles applied to software.

The result is a fully interoperable software library primarily based on the collected software components described in D2.1, with workflows built using such components being executed in a set of popular workflow managers and middleware (workflow manager-agnostic), and in a number of complementary computational environments (infrastructure-agnostic). Besides, the components are described using [CWL](#) and [OpenAPI](#), to improve access and portability, with the added possibility to run them in CWL-compliant workflow managers.

2.2.1 Python Wrappers

BioExcel building blocks were defined as a specific software architecture to contribute to the interoperability, using simple wrappers written in Python to encapsulate software components. The wrappers provide a well-defined interface for input, output, configuration, and provenance. This interface can be fully described and specified using accepted standards like OpenAPI or CWL.

Each of the wrappers provides the necessary format conversions for input and output, and launches the tool inside, which runs unaltered. It can be a command-line tool, a piece of native Python code or a web-service call, and it can also be encapsulated in a virtual machine (VM), in a software container or be executed remotely in an HPC system. These options remain internal to the wrapper and do not affect the external interface, and hence the interoperability. Besides, updates to the inner software tool require only updating the wrapper, maintaining compatibility with previous versions. In order to make the architecture environment agnostic, the wrapper can be called natively from Python, or as a command-line tool.

With an eye toward reducing the possible weight of the downloadable modules, and in an effort to isolate the external tools dependencies, the set of wrappers implemented was divided into different categories, according to the functionality behind the software wrapped. The available biobb modules as of today are:

- [biobb common](#): base package required to use the rest of the biobb packages.
- [biobb io](#): module collection to fetch data to be consumed by the rest of the building blocks.
- [biobb model](#): module collection to model protein/nucleic acids structures (mutations, missing residues/loops) and run structure quality control (checking).
- [biobb md](#): module collection to perform molecular dynamics simulations using GROMACS[11] package.

- **biobb analysis:** module collection to perform Molecular Dynamics Trajectory Analyses using GROMACS package.
- **biobb vs:** module collection to perform pocket detection and docking calculations.

Availability of these modules and how they can be downloaded and used is extensively described in section 2.3.

The number of tools wrapped inside every collection as well as the number of modules available for the biobb package is expected to increase during the last period of BioExcel and during the follow-on BioExcel-2 project. Moreover, the software library is open to allow source contributions, as the wrapper source code is available from the [BioExcel GitHub repository](#). Contributions should follow BioExcel software development best practices presented in the [BioExcel Whitepaper on Scientific Software Development](#).

2.2.2 Workflow manager-agnostic philosophy

The software library described in the previous section was designed to be as compatible as possible with different workflow managers. The only requirement may be the need of a small layer on top of the building block execution, to adapt it to the specific manager needs. All our biobb modules already contain such layers for PyCOMPSs[12] and Galaxy[3] platforms. Moreover, the CWL specification gives the compatibility needed to run them with CWLEXEC, *cwl-runner* or *cwltool* (reference implementations for CWL runs, see [commonwl.org](#)) and any other workflow manager compatible with CWL. KNIME compatibility layers are currently being produced, aiming for the new biobb nodes to be directly available from the KNIME node repository. Jupyter notebooks can be used without the need of any additional layer. Examples of them are prepared for each of the biobb modules, and made available in the corresponding GitHub repository.

Below are examples of how the BioExcel building blocks can be used in a variety of workflow managers:

- **PyCOMPSs:** A new task decorator should be defined, with a set of directives identifying input and output files. Inside the decorated function, just the corresponding launch for the tool is needed, encapsulated in a try/except code block. All BioExcel building blocks have corresponding PyCOMPSs task prepared. Examples can be found here: https://github.com/bioexcel/biobb_md/blob/master/biobb_md/pycompss/gromacs

```

1 from pycompss.api.task import task
2 from biobb_common.tools import file_utils as fu
3 from gromacs import grompp
4
5 @task(input_gro_path=FILE_IN, input_top_zip_path=FILE_IN, output_tpr_path=FILE_OUT)
6 def grompp_pc(input_gro_path, input_top_zip_path,
7             output_tpr_path, properties, **kwargs):
8     try:
9         grompp.Grompp(input_gro_path=input_gro_path, input_top_zip_path=input_top_zip_path, output_tpr_path=output_tpr_path
10                    properties=properties, **kwargs).launch()
11     except Exception:
12         traceback.print_exc()
13         fu.write_failed_output(output_tpr_path)

```

Fig. 2.1 – PyCOMPSSs compatibility layer for the biobb_md:grompp tool.

- CWL-runner, cwltool:** All of the Bioexcel building blocks have been specified in CWL. (https://github.com/bioexcel/biobb_md/tree/master/biobb_md/cwl/gromacs). No extra layer is needed when using these workflow managers, as they are prepared to consume a CWL specification file. BioExcel building blocks use one of these tools for their CWL unitest: https://github.com/bioexcel/biobb_md/blob/master/biobb_md/cwl/test/cwl_unitest.sh
- Jupyter notebooks:** No extra layer is needed to use BioExcel building blocks with jupyter notebooks, as Jupyter has direct support for Python language. However, an example of how they can be executed is prepared for each of the building blocks. Example: https://github.com/bioexcel/biobb_md/blob/master/biobb_md/notebooks/gromacs/pdb2gmx.ipynb
- Galaxy:** A new compatibility layer must be added to use BioExcel building blocks in the Galaxy platform. In this case, an XML-formatted file is needed, containing a description of the bb, dependencies, command line usage, inputs, outputs and help documentation. Example: https://github.com/bioexcel/biobb_galaxy/blob/master/biobb_io/mmb_api/pdb.xml

```

1 <tool id="MMBpdb" name="Fetch PDB File (MMB)" version="0.1.0">
2 <description>Fetch PDB File (MMB) from RSCB PDB code in the conf_file</description>
3 <requirements>
4 <requirement type="package" version="0.0.1">biobb_io</requirement>
5 </requirements>
6 <command>pdb.py --conf_file $conf_file --system galaxy --step $step --output_pdb_path $output_pdb_path</command>
7 <inputs>
8 <param name="conf_file" type="data" format="yaml" label="Configuration file"/>
9 <param name="step" type="text" label="Step Label"/>
10 </inputs>
11 <outputs>
12 <data format="pdb" label="output_pdb_path_label.pdb" name="output_pdb_path"/>
13 </outputs>
14
15 <help>
16 This tool fetch a PDB structure from MMB PDB API
17 </help>
18
19 </tool>

```

Fig. 2.2 – Galaxy compatibility layer for the biobb_io:pdb tool.

- **KNIME:** KNIME analytics platform already offers a set of Python integration nodes ([Python extension](#)), which can be used to run our building blocks just inserting the Python code of the wrappers. However, this implies input and output adaptation, as Python integration nodes in KNIME work using [pandas dataframes](#). Moreover, input parameters must be directly modified in the inserted Python code.

A second approach is the building of native java nodes. As the KNIME platform is a java-based environment, in this case the implementation of the new layers requires a complete installation and configuration of a KNIME-Eclipse Standard Development Kit (SDK). For each building block, a set of java files needs to be written, the node dialog, the node factory, the node model and the node plugin, as well as an XML-formatted file containing a description of the bb, dependencies, command line usage, inputs, outputs and help documentation (similarly to the previous Galaxy case). Possible commercialization opportunities are being explored for these KNIME nodes.

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <knimeNode icon="./biobb.png" type="Source" xmlns="http://knime.org/node/v2.8" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3   <name>PDB_API</name>
4
5   <shortDescription>PDB loader from MMB API PDB mirror (http://mmb.irbbarcelona.org/api/pdb/)</shortDescription>
6
7   <fullDescription>
8     <intro>PDB loader from MMB API PDB mirror (http://mmb.irbbarcelona.org/api/pdb/)</intro>
9
10
11     <tab name="PDB API inputs:">
12       <option name="Enter a PDB id:">PDB code to be fetched (e.g. 1agi)</option>
13       <option name="Output file:">File name where the PDB structure will be stored</option>
14     </tab>
15     <!-- possibly more options that can also be grouped by tabs -->
16     <!--
17     <tab name="Standard Options">
18       <option name="short name of first option (like in the dialog)">description of first option</option>
19       <option name="short name of second option (like in the dialog)">description of second option</option>
20     </tab>
21     <tab name="Advanced Options">
22       <option name="short name of first advanced option (like in the dialog)">description of first advanced option</option>
23     </tab>
24     <!--
25   </fullDescription>
26
27   <ports>
28     <!-- <inPort index="0" name="In-Port name">Description of first input port...</inPort>-->
29     <!-- possibly more input ports here-->
30     <outPort index="0" name="PDB">PDB structure</outPort>
31     <!-- possibly more output ports here-->
32   </ports>
33
34 </knimeNode>

```

Fig. 2.3 – KNIME compatibility layer (xml description file) for the *biobb_io:pdb* tool.

2.2.3 Infrastructure-agnostic philosophy

BioExcel workflows software library was designed to be compatible with different infrastructures. Preliminary benchmarks were presented in D2.2 and D2.3, with executions of the *Model Protein Mutants* workflow in three different environments: single desktop workstation (with different hardware specifications), private cloud (Virtual Machines run in an OpenNebula cloud infrastructure, with different number of cores and memory), and HPC supercomputers, CPU-based ([Marenostrum IV](#), 128 cores) and GPU-based ([Minotauro](#)). The benchmark has been extended incorporating a public cloud infrastructure (EGI), and different combinations of infrastructures and workflow

managers. The simulation length has also been extended to 10ns, reproducing a real equilibration process of a small protein. Table 2.1 presents the different combinations tested.

Infrastructure
Workstation Serial (8 cores, 12GB RAM)
VM Virtual Box Serial (4 cores, 6GB RAM)
VM Open Nebula PyCOMPSs (16 cores, 16GB RAM)
VM Open Nebula Galaxy (2 cores, 8GB RAM)
Marenostrum (CPUs) PyCOMPSs (2 nodes, 16+16 cores, 32 GB RAM)
Minotauro (GPUs) Serial (1 node, 2 GPUs)
Minotauro (GPUs) PyCOMPSs (2 nodes, 2 GPUs)
EGI Serial (1 VM, 16 cores, 16GB RAM)
EGI PyCOMPSs (2 VMs, 16+16 cores, 16+16GB RAM)

Table 2.1 – Model Protein Mutants tests in different infrastructures and combinations of infrastructure and workflow manager.

The same workflow was run in the Marenostrum IV supercomputer, using different number of processors, to test the scalability. Jobs were run using PyCOMPSs workflow manager, and the number of mutations was increased according to the number of nodes used (e.g. 16 nodes → 16 mutations). Results showed a linear speed-up, regardless of the number of nodes used in parallel.

# Cores	Time (hours)
2 nodes = 96 procs	6,06
4 nodes = 192 procs	6,17
8 nodes = 384 procs	6,17
16 nodes = 768 procs	6,18
32 nodes = 1536 procs	6,17

Table 2.2 – Model Protein Mutants Marenostrum IV Benchmarking

Running the workflow in different infrastructures was possible with just adjusting the configuration files according to the environment: binary locations, workflow temporary files path, or input file path can change. The YAML file used as input for our workflows (and also building blocks), described in [D2.2, section 2.2.2](#), supports as many configurations as needed, named “systems” (Fig. 2.4). Example:

https://github.com/bioexcel/pymdsetup/blob/master/workflows/conf/conf_pyruvateKinase_MN.yaml

Different HPC centers and architectures are also being tested (ARCHER – EPCC, Juron – Jülich, CTE-POWER – BSC).

```
41 open_nebula:
42   gmx_path: /usr/local/gromacs/bin/gmx
43   scwr14_path: /home/user/scwr14/Scwr14
44   workflow_path: /home/user/pymdsetup/test_10ns
45   gnuplot_path: /usr/bin/gnuplot
46   # Path to the initial pdb structure (if this field is empty
47   # pdb_code field will be used to download the structure)
48   initial_structure_pdb_path: /home/user/structure.pdb
49
50 mare_nostrum:
51   gmx_path: /gpfs/home/bsc23/bsc23210/gromacs/bin/gmx
52   scwr14_path: /gpfs/home/bsc23/bsc23210/scwr14/Scwr14
53   workflow_path: /gpfs/scratch/bsc23/bsc23210/test_pyruvate
54   gnuplot_path: gnuplot
55   # Path to the initial pdb structure (if this field is empty
56   # pdb_code field will be used to download the structure)
57   initial_structure_pdb_path: /gpfs/home/bsc23/bsc23210/pymdsetup/test/data/PK_Gromacs_Setup.pdb
58
59 minotauro:
60   gmx_path: /gpfs/home/bsc23/bsc23210/gromacs/bin/gmx
61   scwr14_path: /gpfs/home/bsc23/bsc23210/scwr14/Scwr14
62   workflow_path: /gpfs/home/bsc23/bsc23210/pymdsetup/test_10ns
63   gnuplot_path: /gpfs/apps/NVIDIA/GNUPLOT/4.6.3/bin/gnuplot
64   # Path to the initial pdb structure (if this field is empty
65   # # pdb_code field will be used to download the structure)
66   initial_structure_pdb_path: /gpfs/home/bsc23/bsc23210/structure.pdb
67
68 archer:
69   gmx_path: gmx
70   scwr14_path: /home/d118/d118/andrio/scwr1/Scwr14
71   workflow_path: /home/d118/d118/andrio/pymdsetup/test_10ns
72   gnuplot_path: gnuplot
73   # Path to the initial pdb structure (if this field is empty
74   # pdb_code field will be used to download the structure)
75   initial_structure_pdb_path: /home/d118/d118/andrio/structure.pdb
```

Fig. 2.4 – System configurations in the Model Protein Mutants YAML input file.

2.2.4 Best practices: FAIR principles

The [ELIXIR](#) project is working to put in place a series of recommendations to organize the life science bioinformatics ecosystem, establishing Europe-wide standards that can be used to describe life science data. Recently, the FAIR guiding principles of data management [9], has started to be extended to

software, as the basic requirements (registries, standards, unified software managers) were already identified and available. In this context, recommendations to encourage best practices in research software were collected and presented (Open Source Software-OSS), aligning the data FAIR principles to this software recommendations[10]. The BioExcel software library has been implemented in a manner compatible with the FAIR principles. The strong collaboration between ELIXIR interoperability platform and BioExcel underpins the FAIR principles of interoperability and reusability. This has been presented in different European conferences ([ISOBP](#), [PASC](#), [Nettab](#)), [posters](#), and a draft paper showing the work done in this context is being prepared, to be submitted in the F1000 research ELIXIR channel.

- **Findability**

The primary requirement for findability in the case of software is availability in a software registry. Traditional software repositories like GitHub and others are suitable for such usage although they are not designed as data resources, and the amount of scientific metadata is limited. ELIXIR has pushed for the establishment of its own tools registry ([bio.tools](#) [13]). Bio.tools includes a large set of metadata that makes tools findable according to their scientific utility, provides extended metadata regarding publication, documentation and support. In addition, it is linked to a software benchmarking platform, [openEBench](#) [14], that in turn provides software technical and scientific quality data. bio.tools has assigned a unique identifier to the registered tools, and these identifiers could be resolved either at bio.tools itself, openEBench, and also the recommended ELIXIR site, [identifiers.org](#).

One of the most attractive features of bio.tools is the use of an extended ontology ([EDAM](#) [15]) to annotate tools. EDAM annotation allows to classify tools according the type of data they consume or produce, and to have a controlled vocabulary to define their precise functionality. In turn, this information can be used to automatically discover new tools consuming a particular input type. Unfortunately, ontology terms for structural bioinformatics are scarce in EDAM, in particular for biomolecular simulation.

BioExcel partners did a first attempt to register the biomolecular tools collected in the D2.1, and this exercise uncovered a number of missing data types, file formats, and functionalities related to biomolecular simulations. Those absent fields were gathered, described in an EDAM-like format, and were [proposed as a potential addition](#) to the ontology. The additions include simulation related operations like *force-field parameterization*, *essential dynamics* or *structure visualization*, specific data types like *system topology*, *simulation trajectories*, or *simulation principal components*, and file formats covering the most popular simulation codes such as *XTC*, *TNG*, *ITP* (GROMACS), *PSF*, *BinPos* (NAMD) or *OFF*, *frmod* (AMBER). These new terms are being included in EDAM, to be released by late 2018.

There are currently 33 tools registered in bio.tools within the [BioExcel collection](#). These correspond to the identified biomolecular tools gathered in

D2.1 (26 curated entries), plus a number of newly added registries, corresponding to the library of building blocks and the workflows implemented during the lifetime of the project, including those behind the pilot use cases. The number of tools registered in this collection is expected to increase during the BioExcel-2 period.

- **Accessibility / Availability**

Following OSS and software packaging and containerization recommendations[10, 16], we have put particular effort in making the BioExcel workflow building blocks accessible/available. Python code is available through the [BioExcel github](#) repository. [BioConda](#) packages and [Biocontainers](#) are being built from them (see section 2.1.2), and Virtual Machines with all dependencies previously installed can be already deployed from the [BioExcel cloud portal](#) and EGI AppDB BioExcel VO (see section 4.2.2). All packages are being prepared for Galaxy toolshed[3, 17] and KNIME[4] node repository. Pre-configured [environment modules](#) are being prepared and installed in HPC centers such as the Barcelona Supercomputing Center (BSC), and will be exported to other centers (e.g. EPCC, Jülich).

- **Interoperability**

The theoretical recipe for full tool interoperability is simple: the use of a common data model. Unfortunately, attempting to generate a single data model for bioinformatics has aroused to be a hard issue [3, 8]. Our approach was to define a specific software architecture to contribute to the interoperability: the Python wrappers introduced in section 2.1.1.1. The selection of Python language, currently the n°1 in the list of programming languages popularity (<http://pypl.github.io/PYPL.html>), makes the library easily accessible, shared, and broadly applicable. The common way of *consuming* (input files + configuration) and *producing* (output files + log) data ensures modularity, flexibility and interoperability (see [D2.2](#) for an extended description). The new biomolecular simulation terms in EDAM ontology will also be a step forward to the introduction of a standard vocabulary in the field. Our library supports interoperability in its more technological point of view, i.e. use of heterogeneous systems with minimal changes[18], as it described above.

- **(Re)usability**

BioExcel building blocks and workflows are all accompanied by a Common Workflow Language[19] (CWL) description, which ensures reproducibility (see next section). Python and CWL blocks have unit-testing modules defined. Software technical documentation is automatically generated using [readthedocs](#). Comments are written within the code following [docstrings](#) conventions, and are transformed afterwards to html and markdown using [sphinx](#). Programmatic access and API documentation and metadata will be available through [openAPI](#) specification.

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 permit redistribution, reuse and repurposing, as long as the attributions and original license are retained. The Apache License is thus often considered business-friendly, as it allows commercial exploitations and customizations, which we think are essential for workflow building blocks.

Note however that the underlying software, which these wrappers effectively call as command line tools, are distributed under several open source licenses, for instance GROMACS is distributed under [LGPL 2.1](#) which would independently govern any commercial changes to the GROMACS binary as opposed to the BioExcel wrapper.

2.2.5 Common Workflow Language

The [Common Workflow Language](#) (CWL)[19] is a specification for describing workflows and tools that are portable and scalable across a variety of software and hardware environments, from workstations to cluster, cloud, and high performance computing (HPC) environments. CWL has been adopted by the ELIXIR project as the recommended standard language to describe workflows.

From the point of view of BioExcel users, the benefits of CWL interoperability are twofold: Firstly, CWL provides a *structural* description of command line tools (e.g. input/output file types, parameters and options) that are reusable across workflow engines. Secondly, as CWL workflows are *portable* across multiple workflow engines, BioExcel-produced workflows described in CWL can be executed in multiple workflow managers. Extended [information about BioExcel's use of CWL](#) was included in D2.2.

All BioExcel building blocks are described using CWL. Each of them has a CWL file with three main fields specified: *base command*, *inputs* and *outputs* (see Fig 2.5). Base command tells CWL which command line needs to be used to run the particular building block; inputs and outputs define the number, name, type, and position and prefix in the command line execution. Parameters previously specified in the CWL file can be referenced using $\$()$ notation to reference input parameters in other fields, e.g. $\$(inputs.output_gro_path)$ or $/$ notation to reference external input files, e.g. `mutate_structure/scw_output_pdb_file` (http://www.commonwl.org/user_guide/06-params/).

Fig. 2.5 shows an example for a simple building block, *pdb2gmx*, included in the [biobb_md](#) module, running the first step of a Molecular Dynamics preparation with GROMACS, generating a topology from a PDB structure. In the figure, section *a* displays the CWL *pdb2gmx* building block specification, with 6 different inputs and 2 outputs. Section *b* shows an example of how this CWL specification can be integrated in a more complex workflow, the protein mutations workflow (*pymdsetup*, extensively described in D2.2), here shortened for the sake of clarity.

```

a) 1 #!/usr/bin/env cwl-runner
    2 cwlVersion: v1.0
    3 class: CommandLineTool
    4 baseCommand:
    5   - pdb2gmx.py
    6 inputs:
    7   system:
    8     type: string
    9     inputBinding:
   10      position: 1
   11      prefix: --system
   12      default: "linux"
   13   step:
   14     type: string
   15     inputBinding:
   16      position: 2
   17      prefix: --step
   18      default: "pdb2gmx"
   19   conf_file:
   20     type: File
   21     inputBinding:
   22      position: 3
   23      prefix: --conf_file
   24   input_pdb_path:
   25     type: File
   26     inputBinding:
   27      position: 4
   28      prefix: --input_pdb_path
   29   output_gro_path:
   30     type: string
   31     inputBinding:
   32      position: 5
   33      prefix: --output_gro_path
   34   output_top_zip_path:
   35     type: string
   36     inputBinding:
   37      position: 6
   38      prefix: --output_top_zip_path
   39 outputs:
   40   output_gro_file:
   41     type: File
   42     outputBinding:
   43      glob: ${inputs.output_gro_path}
   44   output_top_zip_file:
   45     type: File
   46     outputBinding:
   47      glob: ${inputs.output_top_zip_path}

b) 1 #!/usr/bin/env cwl-runner
    2
    3 cwlVersion: v1.0
    4 class: Workflow
    5
    6 inputs:
    7   scw_input_pdb_path: File
    8   mutation: string
    9
   10 outputs:
   11   gnuplot_output_png_file:
   12     type: File
   13     outputSource: gnuplot/gnuplot_output_png_file
   14   rms_output_xvg_file:
   15     type: File
   16     outputSource: rmsd/rms_output_xvg_file
   17   md_output_gro_file:
   18     type: File
   19     outputSource: equilibration/md_output_gro_file
   20   md_output_trr_file:
   21     type: File
   22     outputSource: equilibration/md_output_trr_file
   23
   24 steps:
   25   mutate_structure:
   26     run: scw1.cwl
   27     in:
   28       scw_input_pdb_path: scw_input_pdb_path
   29       scw_mutation: mutation
   30     out: [scw_output_pdb_file]
   31   create_topology:
   32     run: pdb2gmx.cwl
   33     in:
   34       p2g_input_structure_pdb_path: mutate_structure/scw_output_pdb_file
   35     out: [p2g_output_gro_file, p2g_output_top_zip_file]

c) 1 #!/usr/bin/env bash
    2
    3 BIOBB_MD=$HOME/projects/biobb_md/biobb_md
    4 cwltool $BIOBB_MD/cwl/gromacs/pdb2gmx.cwl $BIOBB_MD/cwl/test/gromacs/pdb2gmx.yml

d) 1 #!/usr/bin/env bash
    2
    3 BIOBB_MD=$HOME/projects/biobb_md/biobb_md
    4 cwltool $BIOBB_MD/cwl/gromacs/pdb2gmx.cwl $BIOBB_MD/cwl/test/gromacs/pdb2gmx.yml

```

Fig. 2.5 – CWL example for [biobb md:pdb2gmx](#) building block (bb). a) bb specification using CWL; b) workflow example using the bb; c) configuration file to run the bb; d) unit-test to run and test the bb

The second step of the workflow corresponds to the *pdb2gmx* building block, consuming as input a PDB file coming from the previous step (mutation modeling), and producing as output two different files: a GROMACS gro file and a zip file containing all the files needed to represent the structure topology in GROMACS: top + itp files.

Section c displays a configuration file (YAML) used to run the *pdb2gmx* CWL, where inputs and outputs files are designated. This configuration file, together with the CWL description of the building block (Fig. 2.5a) is used in a unit-test validation for every single biobb unit.

Section *d* represents an example of the unit-test for this particular bb. All these files (CWL specifications, YAML configuration files, SH unit-tests) can be found in the BioExcel github.

As it can be seen in Fig2.5b, a complete workflow specification in CWL is built from putting together different CWL descriptions, and joining them by binding outputs from a particular step to corresponding inputs of the following step. Then, the complete CWL-described workflow can be run with just a single command line, using *cwl-runner* or *cwltool* (reference implementations for CWL runs, see commonwl.org) or any other workflow manager compatible with CWL. An example of a completely CWL-described workflow is the *pymdsetup*: [cwl file](#), [yaml configuration file](#), [unit-test sh](#).

Moreover, workflows described in CWL can be directly transformed into a graphical diagram, thanks to the work of BioExcel partner The University of Manchester: The CWL Viewer (<https://view.commonwl.org/>) is a richly featured web visualization suite, which graphically presents and lists the details of CWL workflows with their inputs, outputs and steps [[10.7490/f1000research.1114375.1](https://doi.org/10.7490/f1000research.1114375.1)]. It also packages the CWL files into a downloadable Research Object Bundle[20] including attribution, versioning and dependency metadata in the manifest, allowing it to be shared easily. The tool operates over any workflow held in a GitHub repository. Other features include: Path visualization from parent and child nodes; nested workflows support; workflow diagram download in a range of image formats; a gallery of previously submitted workflows; and support for private Git repositories and public GitHub including live updates. Examples of workflows generated by BioExcel can be seen here:

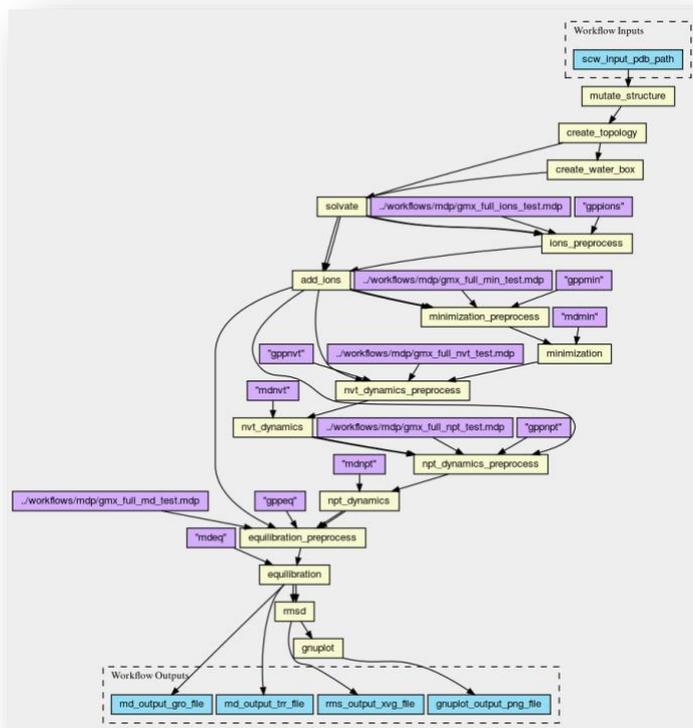


Fig. 2.6 – CWL Viewer diagram for the [Modeling Protein Mutants Workflow](#)

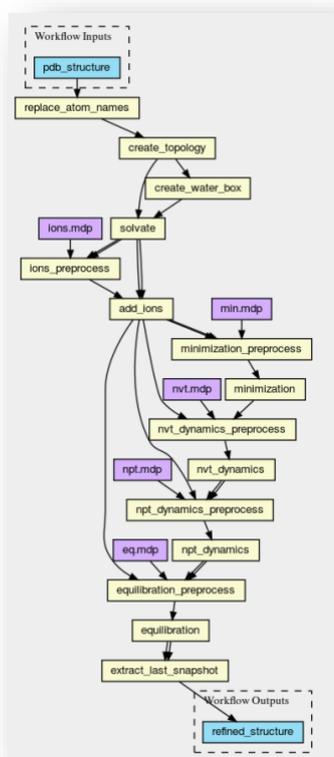


Fig. 2.7 – CWL Viewer diagram for the [Protein-protein energy refinement](#) (work together with [MuG](#) project)

2.3 Availability

BioExcel workflows software library has been designed to reach a broad number of biomolecular fields, with different users' level: expert users, those that want to squeeze the maximum from biomolecular simulations, primarily working using supercomputers from HPC centers; entry level users, those who are interested in the field, but are scared (or just unable) to start using the available tools; and intermediate users, those who are interested in biomolecular simulations and know how to use the technology behind, but don't want to spend time installing, configuring or tuning the tools, preferring the ease of use rather than the performance. BioExcel has been working towards different approximations, trying to cover the whole users' diversity.

2.3.1 Expert users (command line, HPC)

BioExcel as a center of excellence has a direct contact with expert users. Complex and advanced biomolecular studies, involving computer-demanding tasks, can be advised and helped by the range of experts in molecular simulations (GROMACS, HADDOCK, CPMD) and HPC (EPCC, BSC, KTH) that are partners of the project. Consequently, the first efforts in the design and development of biomolecular workflows and building blocks were focused on massively parallel workflows, and thus, with the possibility to bind HPC workflow managers with our software library. Always with an eye put on the future exascale systems, this approach is currently dealing with a technical success case using up to 40,000 different processors at the same time with the model protein mutants workflow, after successfully run the first test with ~10,000 processors (see section 2.1.3). In order to make this particular workflow (and all HPC-intensive BioExcel workflows) portable and easy to use in different supercomputers, a module environment will be prepared for each of them. This module environment will load all the dependencies needed by the workflow to be successfully executed on a determined HPC supercomputer. Modules add transparency to the software being executed by the workflow, which may change depending on the system architecture (e.g. GPU vs CPU-based supercomputers).

As presented in the previous sections, BioExcel building blocks and workflows are written in Python language. The source code is all available from the [BioExcel GitHub repository](#). Installation and configuration of all the software needed by the different blocks is described in step-by-step README files. Workflows available from the repository have been tested in general Unix-based machines, and should run properly in either a Linux machine as well as in local cluster infrastructures.

gromacs.pdb2gmx module

```

class gromacs.pdb2gmx.Pdb2gmx(input_pdb_path, output_gro_path, output_top_zip_path, properties,
                               **kwargs) [source]
Bases: object
Wrapper class for the GROMACS pdb2gmx module.

Parameters:
  • input_pdb_path (str) - Path to the input PDB file.
  • output_gro_path (str) - Path to the output GRO file.
  • output_top_zip_path (str) - Path to the output TOP topology in zip format.
  • properties (dict) -
    - output_top_path (str) - ("p2g.top") Path of the output TOP file.
    - output_itp_path (str) - ("p2g.itp") Path of the output itp file.
    - water_type (str) - ("spce") Water molecule type. Valid values: tp3p, spce, etc.
    - force_field (str) - ("amber99sb-ildn") Force field to be used during the
      conversion. Valid values: amber99sb-ildn, opisaa, etc.
    - lghn (bool) - (False) Should pdb2gmx ignore the hydrogens in the original
      structure.
    - gmx_path (str) - ("gmx") Path to the GROMACS executable binary.

```

Launch() [source]
Launches the execution of the GROMACS pdb2gmx module.

Fig. 2.8 – Documentation example for the `biobb_md:pdb2gmx` building block. Descriptions of the different available input parameters for the particular building block are shown. Prepared using Read the Docs.

Developers working in the biomolecular simulation field can program their own workflows in Python using BioExcel building blocks. The libraries are available to download and install from the Python Package Index ([PyPI](#)) and BioConda package manager ([BioConda](#)). Configuration parameters for each of the building blocks can be looked up at the documentation prepared using Read the Docs (e.g. Fig.2.8). Links to all available repositories and documentation are listed in the Appendix Section.

Programmers can exploit the interactivity offered by the [Jupyter notebooks](#) in Python. Thanks to the modularity of the building blocks, a workflow can be assembled step by step, and the intermediate results can be checked before going ahead with the complete execution, giving the possibility to change or tune the input parameters used. Once the desired execution and results are obtained, the Python workflow can be exported to production.

```

# Import module to test
from gromacs.pdb2gmx import Pdb2gmx

# Create prop dict and inputs/outputs
prop = {}
input_pdb_path = '../test/data/gromacs/pyruvate_kinase.pdb'
output_gro_path = 'output_gro.gro'
output_top_zip_path = 'output_top_zip_path.zip'

# Create and launch bb
Pdb2gmx(input_pdb_path=input_pdb_path, output_gro_path=output_gro_path, output_top_zip_path=output_top_zip_path, prop=prop)

# Show log
!head -n 3 log.out
2018-10-10 16:17:11,370 [MainThread ] [INFO ] gmx pdb2gmx -f ../test/data/gromacs/pyruvate_kinase.pdb -o output_gro.gro -p p2g.top -water spce -ff amber99sb-ildn -i p2g.itp
2018-10-10 16:17:11,370 [MainThread ] [INFO ] Exit code 0

# unzip list
from biobb_common.tools.file_utils import unzip_list
file_list = unzip_list(output_top_zip_path)

# Always run ngview with Mozilla Firefox
# Might need to run jupyter-orientation enable ngview --py --sys-prefix
# Therefore launch jupyter-notebook in order to enable ngview command
# Visualize result with ngview
import ngview
view = ngview.show_file(output_gro_path)
view

```

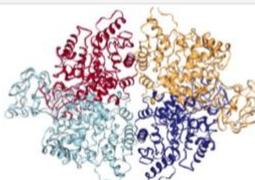


Fig. 2.9 – Jupyter notebook example for the `biobb_md:pdb2gmx` building block

2.3.2 Entry-level users (GUI)

BioExcel has always thought about the entry-level users. As a center of excellence, we want to be an entry point for all scientists interested in biomolecular simulations, to ease with the typically hard starting phase. For that reason, an [entry-level users interest group](#) was created. The main commonality of this group of users is the inclination towards a graphical and intuitive way of running the tools rather than a command-line way. BioExcel worked hard during the design phase to make our building blocks usable in different environments, with special focus on *how* to run them. Thanks to that, there are now three different approaches going on porting the software library to graphical user interfaces: Galaxy, KNIME and a dedicated web server.

The Galaxy workflow manager[3], already presented in D2.1 and D2.2, is an open, web-based platform for data intensive biomedical research. Galaxy works building a workflow implicitly by applying a series of operations on the data items, keeping a history of all intermediate data items that are produced (and how they were made), making it easy to rerun parts of the workflow and share the results with others. Galaxy is tightly integrated with a large collection of tools for genomics and sequence analysis (Galaxy toolshed[17]), and is therefore popular for Next-Gen Sequencing data analysis. However, it is not so well-known in the structural biomolecular field, mainly due to its lack of tools focused on structures. BioExcel bb were successfully integrated in a local Galaxy infrastructure (see D2.2), however, we identified a set of issues mainly related to the difficult installation of all the dependencies needed. These issues could be solved thanks to the recent packaging of the building block modules using the Anaconda package manager and the BioConda package repository. Galaxy is compatible with BioConda packages, and is able to automatically download and install all packages needed for a particular package being installed. Currently, the whole BioExcel library is being ported to the Galaxy toolshed. This work is bringing us closer to the genomics and sequence analysis specialists, increasing our visibility and attention.

During the discussions between BioExcel WP2, WP3 and WP5, a particular need for a workflow approach better suited to pharmaceutical companies was identified. Consequently, new efforts were put to the porting of our software library to KNIME: an open source workflow system, with an easy and intuitive drag and drop based Graphical User Interface (GUI), very popular in cheminformatics for data analysis, statistics and visualization, and heavily used by pharmaceutical companies. Its functionality is based on a set of modules (KNIME nodes) for data integration, which can be interconnected, generating custom pipelines (workflows). Currently, KNIME has >1500 different free nodes, which are organized in sections, each one dedicated to a special task: data access, data manipulation, data analytics, etc. However, the list of available nodes in the GUI can be extended importing KNIME extensions. In a similar way to the previously presented Galaxy workflow manager, which is biased to genomic tools, KNIME list of nodes is biased to cheminformatics and data analytics.

New BioExcel KNIME nodes for our building blocks (biobb) are being generated, building a new biobb category in the KNIME Node Repository. That implies the corresponding java code wrapping the building block, and a complete description for each of them, which is going to be shown in the Node Description section of the GUI. The set of new developed nodes will be then available for KNIME users to generate their own biomolecular simulation workflows, using not only BioExcel biobb nodes, but also the set of nodes (>1500) already available in the Node Repository, including the [chemistry related ones](#) ([RDKit](#), [Marvin](#), [Vernalis](#), [Schrödinger](#), [3D-e-Chem](#)). In a similar way than the work with Galaxy, this work is bringing us closer to the pharmaceutical field, also increasing our visibility and attention.

In addition to the previous work, the discussions between BioExcel WP2, WP3 and WP5, also identified a need to have an independent, dedicated entry point where users could find all information regarding BioExcel building blocks and software library. According to that, a new Graphical User Interface (GUI) started to being developed: the biobb web server, with the idea of ease the use of BioExcel workflows and tools for entry-level users, but also advertise the many different ways in which the library can be used. This will complement the BioExcel Workflow Repository mini-project being developed in WP3 (D3.6), which will promote the workflow definitions for reuse and repurposing by more proficient workflow system users.

On the tools part, biobb web server will allow users to run a set of chosen, pre-configured workflows composed from BioExcel building blocks, such as a structure quality checking, a structure energy minimization, a complete MD setup, or a complete MD simulation (with length restrictions). The GUI will also provide an additional interactivity to our building blocks. A great example is the possibility to run a quality check of a structure, while at the same time a 3D representation of the molecule is shown in the same interface, highlighting the region of the structure of particular interest. This interactivity can be applied also to the set of analyses generated by the workflows. Workflows will be submitted and treated by a queue manager, which will serve them in an on-demand processing model performed by Virtual Machines, automatically deployed in an Open Nebula OneFlow cloud environment. A direct connection to HPC supercomputing infrastructures to submit long molecular dynamics simulations prepared using the portal will be studied.

On the documentation part, the web page will gather all the information on how to obtain, install and run the building blocks and workflows generated by BioExcel: for developers or experts in the field ([github](#), [bioconda](#), [biocontainers](#), [VMs/Cloud](#)) for HPC users ([environment modules](#)) and for entry-level users ([Galaxy](#), [KNIME](#), web server).

2.3.3 Intermediate users

BioExcel plans have always taken into consideration those users that are in-between the entry-level and the experts, more interested in being able to run

a particular simulation in a quick and easy way, without having to install or configure anything in its own premises, in spite of the performance cost. Pre-packaged software into Virtual Machines or software containers has been prepared and a BioExcel cloud portal, able to deploy these images, is already working (see section 4.2). Besides, a BioExcel Virtual Organization was created in the EGI AppDB (presented in D2.2 and D2.3), which allows registering of our Virtual Machines as Virtual Appliances, that can be then deployed using the BioExcel Cloud portal or one of the cloud providers shared with ELIXIR VO.

2.4 Technical success story: Pyruvate Kinase annotated mutations

The transversal workflow “*Model Protein Mutants*”, extensively presented in the previous D2.2, has been used as a prototype to test the designed workflows development process in the project and also the possibility to run them in different computational infrastructures. A particular implementation of this workflow, using PyCOMPSs as a workflow manager to control the parallel executions, and the Marenostrum IV supercomputer to run the Molecular Dynamics simulations was used to perform a technical challenge: a massive study of variant flexibility for the Pyruvate Kinase protein. Observations extracted from the resulting trajectories can be used in the context of prediction of the pathogenicity of such variants, using structural and dynamics features, coupling structure and flexibility with function.

Pyruvate kinase is one of the most widely studied enzymes throughout the history of biochemistry, due to its major role in the regulation of glycolysis[21]. It catalyzes the irreversible conversion of phosphoenolpyruvate (PEP) to pyruvate, generating an ATP molecule in the process. Pyruvate kinase has a highly conserved architecture throughout evolution. In most organisms, it is a 200-kDa homo-tetramer (Fig. 2.10A), whereas each subunit is generally composed by four well-defined domains (Fig. 2.10B): domain A, B, C and N-terminal. The particular case chosen for this study is the human erythrocyte pyruvate kinase (R-PYK), one of the four known human pyruvate kinase isoforms, encoded by gene *PKLR* and expressed in erythrocytes (574 residues per subunit)[22]. The catalytic center of the enzyme is located in the cleft between domains A and B. R-PYK is allosterically regulated by fructose-1,6-biphosphate (F16BP), an activator of its catalytic efficiency, which binds to the allosteric site located at domain C (Fig. 2.10B).

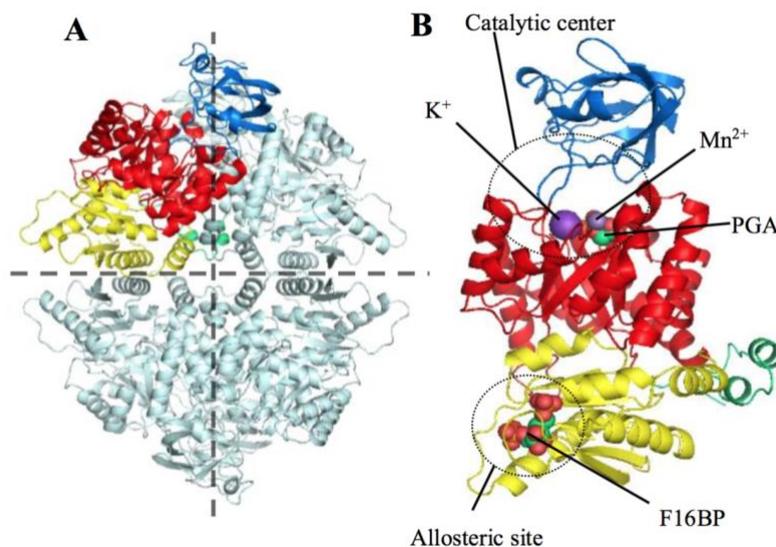


Fig. 2.10 – 3D-representation of the Pyruvate Kinase structure. A) View of the homotetramer; B) View of a monomer subunit, with domains N-terminal, A, B and C colored green, red, blue and yellow respectively. Catalytic center and allosteric site locations are highlighted.

The particular PK case chosen for this study, the human erythrocyte pyruvate kinase, was selected because of its large number of annotated missense variants that have been associated to a disease called nonspherocytic hemolytic anemia, a rare, autosomal recessive disease which causes blood disorders characterized by the premature destruction of red blood cells (erythrocytes), resulting in anemia. A set of 200 mutations consisting on reported pathogenic variants were manually selected and curated from the whole set of variants available at the UniprotKB database (<https://www.uniprot.org/uniprot/P30613>), and used as input for the *Model Protein Mutants* workflow. A diagram of the workflow was presented in D2.2, Fig. 4.1.

PyCOMPSs[12] workflow manager was used to control the massive parallelism. The 200 independent mutations were split into 200 different nodes of the Marenstrum IV supercomputer. This supercomputer, located in the BSC premises and updated in 2017, gives a total performance of 11.15 Petaflops from its 165,888 processors (3,456 nodes with two Intel Xeon Platinum chips, each with 24 processors). Using one node per each MD simulation then, the workflow used 9,600 different processors in parallel, 48 processors per simulation, run with GROMACS package. The workflow was launched in one single job, with 9,600 processors reserved, and produced as result a single RMSd plot, containing the dynamic behavior of the different protein variants, measured by the average distance between the atoms of the computed trajectories and the initial structures. The pipeline used, written in Python, can be found here: https://github.com/bioexcel/pymdsetup/blob/master/workflows/pyruvateKinase_MN.py

Next step on this challenge is trying to port each of the simulations to the MPI regime, using more than one node per simulation. This could increase even one order of magnitude the number of processors able to be used in parallel with

the workflow, although care is needed as the system studied should be big enough to be parallelized (splitting the system) using thousands of processors. In this case, the simulated system is made of more than 500,000 atoms (homotetramer + ions + solvent), so the next test being prepared will use 4 nodes (192 processors) per simulation, adding up to a total of 38,400 processors. With new supercomputers increasing the number of processors (Chinese Sunway TaihuLight has 10,649,600 cores), the possibility to use this large number of cores in parallel with a single workflow is of crucial importance. BioExcel will continue its work towards exascale in the BioExcel-2 period approaching the ideal usage of large HPC systems in this field:

“An ideal approach would for instance be to run a simulation of an ion channel system over 1000 cores in parallel, employing, e.g., 1000 independent such simulations for kinetic clustering, and finally perhaps use this to screen thousands of ligands or mutations in parallel (each setup using 1,000,000 cores)” [23].

2.5 Scientific success story: Epidermal Growth Factor Receptor cancer mutations

One of the pilot use cases proposed in the BioExcel project was the design of a Virtual Screening pipeline, with target (protein) flexibility represented by an ensemble docking (see section 3.5). The part of the pipeline generating the ensemble is currently using the *Model Protein Mutants* (pymdsetup) workflow. Nostrum Biodiscovery, a BSC spin-off working with pharmaceutical and biotech companies, proposed a scientific case to setup and fine-tune the Virtual Screening pipeline: the Epidermal Growth Factor Receptor (EGFR). Studies on the dynamics and flexibility of EGFR variants have already exposed important information, not only about the abnormal function of the protein, but also about the mechanism of resistance against particular drugs. The aim of this challenge is to reproduce these results using our workflow, and proof its reliability towards possible new scientific use cases.

EGFRs are transmembrane receptors located on the cell membrane. They have an extracellular binding domain, to which Epidermal Growth Factor (EGF) binds, a transmembrane domain and an intracellular tyrosine kinase domain (Fig. 2.11). EGFRs are active only after dimerization, stimulating their intrinsic intracellular protein-tyrosine kinase activity. This dimerization is triggered by the Epidermal Growth Factor ligand.

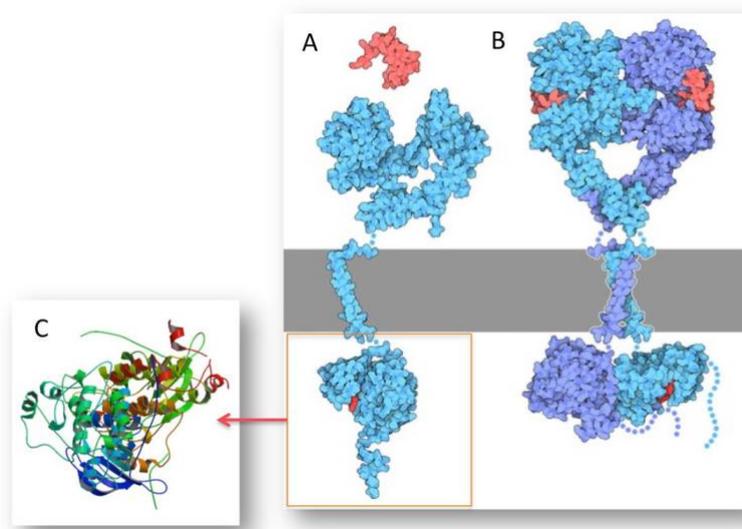


Fig. 2.11 – 3D-representation of the Epidermal Growth Factor Receptor (EGFR). A) Monomeric structure. B) Dimeric structure. Epidermal growth factor is shown in red and its receptor in blue. The extracellular part (top), transmembrane region (grey, middle) and intracellular part (bottom) are shown. C) Intracellular Tyrosine-Kinase subdomain.

EGFRs play an important role in controlling normal cell growth, apoptosis and differentiation. Mutations of EGFRs can lead to abnormal activation and signal transduction causing unregulated cell division and ultimately driving some types of cancers. Thus, dysregulation of EGFR activity has been implicated in the oncogenic transformation of various types of cells and represents an important drug target. Currently, there are two therapeutic approaches hitting EGFR. One of them is based on monoclonal antibodies, which bind to the extracellular domain of the receptor, antagonizing either the interaction with its cognate ligand (EGF) or its homo or hetero dimerization. The second therapeutic approach is knocking down its tyrosine-kinase activity. This is also a very interesting option as there are several therapies with marketing authorization approvals that target its kinase domain. Approved small molecule drugs in this category are ATP competitive inhibitors, either reversible or covalent. Some examples are: *Gefitinib*, *Vandetanib*, *Lapatinib*, *Erlotinib* and *Afatinib*. Importantly, the administration of these treatments imposes a selection pressure on the cancer cells, which eventually develop mutations in the kinase domain that lead to resistance. One of the most prevalent mutations found in treated patients is the T790M mutation. This change is located in the so-called “gatekeeper” residue, in the interior of the ATP binding site. The replacement of a small threonine amino acid for a much bulkier methionine precludes or partially hinders the binding of the ATP competitive treatments listed above. This problem has spawned the development of a next-generation of ATP competitive inhibitors that target the T790M mutant, such as *osimertinib*, *rociletinib*, *HM61713*, *ASP8273*, *EGF816* and *PF-06747775*.

The study on the flexibility and dynamics differences between the variants and the wild type started with a list of 7 mutations known to confer resistance to some of the above-mentioned treatments:

- T790M (gatekeeper) confers resistance to *Erlotinib* and *Gefitinib* by increasing ATP binding.
- L718Q, L844V confer resistance to *Rociletinib*.
- M766T, L858R, L718A, T854A

The Molecular Dynamics simulations were launched using the same approximation presented in the previous technical case: a combination of pyCOMPSs workflow manager with the Marenstrum IV supercomputer. The simulations for the different variants were extended up to 1 μ s. The pipeline used, written in Python, can be found here:

https://github.com/bioexcel/pymdsetup/blob/master/workflows/egfr_md.py

The first analysis to the different mutated protein dynamics show a large flexibility in the ATP binding site, being defined by the residues in contact with the ANP molecule (ATP antagonist) from the PDB code 2ITN (Fig.2.12).

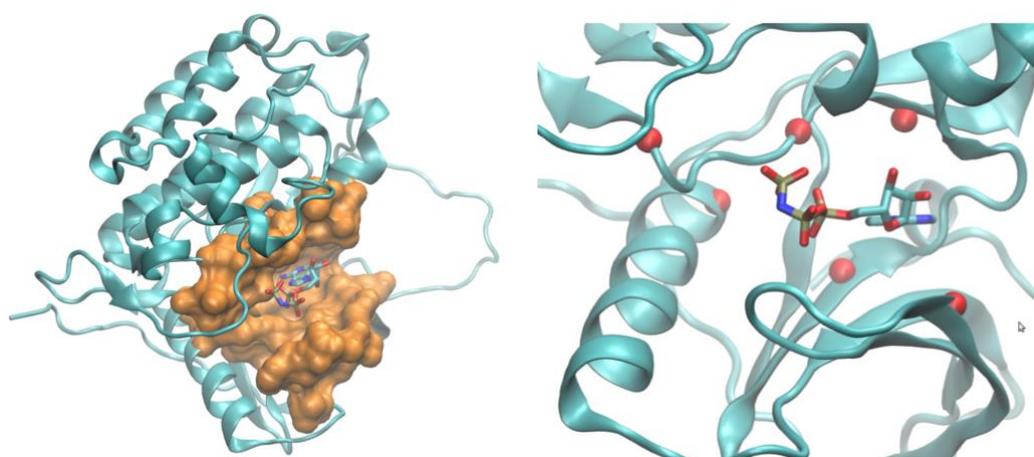


Fig. 2.12 – Left: EGFR kinase domain active site, defined by the residues in close contact (8 Å) from the ligand ANP (ATP antagonist from PDB code 2ITN). Right: Position of the mutated residues used in the study, with ligand ANP shown in Licorise representation.

In particular, the distance between two residues from the active site involved in a crucial salt bridge (Lys50 and Glu67) has been shown to vary a lot depending on the mutation analysed. The salt bridge is known to be present when the kinase is in its active form (when having ATP bound and ready to transfer phosphate to another protein). In 4 out of 7 variants, the salt bridge is broken during the dynamics (Table 2.3).

A large number of analyses are being planned on the active site of the protein, to compare the flexibility behavior of the different variants, such as Solvent Accessible Surface Area (SASA) along time, RMSd and atomic fluctuation along time, and Principal Component Analysis (PCA). With all these analysis together, we expect to reproduce the role played by the flexibility and dynamics of the active site residues in the mechanism of resistance against particular drugs.

Results obtained by this project will be also used for one of the project pilot use cases: Virtual Screening. An ensemble of structures for each of the variants simulated has been computed and will be used in the ensemble docking section of the Virtual Screening, offering important target flexibility information in the docking process.

Variant	Salt Bridge during MD
<i>Wild Type</i>	YES
<i>Leu149Val</i>	YES
<i>Leu163Arg</i>	Partially
<i>Leu23Ala</i>	NO
<i>Leu23Gln</i>	NO
<i>Met71Thr</i>	NO
<i>Thr159Ala</i>	NO
<i>Thr95Met</i>	YES

Table 2.3 – EGFR kinase domain Lys50 - Glu67 salt bridge occurrence during MD simulations of the different variants studied.

3 Pilot Use Cases

3.1 Pilot Use Case 1: High Throughput Workflow for Cancer Genome Sequencing Data

- *Use case summary*

Continuing work as described in deliverable 2.3, we have developed two main workflows for use in rapid turnaround cancer genome analysis on sequencing data from high throughput Next Generation Sequencing (NGS) systems. For each patient, a cancer sample and normal/blood sample is sequenced, producing a pair of reads for each sample. In total, these 4 files can be of the order of several hundreds of GBs in size, with final aligned and indexed files not much smaller in size. Given the projected increase in the use of genome analysis in medical practice, having a pipeline or workflow which can handle this large amount of data robustly, with speed, scalability, and flexibility is vital.

At present, our workflows are kept separate, as discussed below, but it is possible for users to simply create their own scripts (e.g. in Python or Bash) which can automate the workflow from one stage to the other, while also allowing the user to incorporate the tools into their own workflows. We also performed some benchmarking of tools used in these workflows on HPC systems, including estimating the performance gain for new tools that became available towards the later stages of development. The packages and workflows developed as part of this project will be freely available online from late October/early November.

- *Workflows*

The workflow created as part of this project covered two main stages: 'Sequence Quality Control' and 'Alignment'. Our main aims for these workflows were to improve ease of use, robustness, speed and scalability. Each stage consists of installable Python modules, each of which controls a specific step within the workflow, providing access to useful tools for the user. Along with the modules, we also developed a command line tool which can run the entire workflow stage, defined in previous deliverables, with one command when given the appropriate arguments. One benefit of creating the workflows in this way is that each step can be used in other scripts/workflows, providing the user with a great deal of flexibility and is a stark improvement on the original workflow. For example, a user can create their own Python script, which imports certain steps from our packages as a module within a wider, more unique workflow. Each module/step can also be executed independently with the command '`python -m <module.function>`'. This allows the user to incorporate each stage independently into workflow managers such as CWL. Figures of the two stages can be found in D2.3 (Figs 2 and 3).

- *Sequence Quality Control*

The first workflow stage performed a series of quality control steps on genome sequences received by the user from NGS systems. A summary of the quality of the samples is created, which then is used to decide how the workflow will proceed, based on a set of criteria. Currently it is possible for the workflow to perform a series of ‘trims’ on the sequences to remove either poor quality or remaining adapter sequences used by the sequencing machine. A second round of quality control checking and trimming can then also be performed if necessary. This automated checking step is the biggest usability improvement for our partners, as these quality checking and trimming stages used to be performed manually. By removing this stage from user intervention, the same pipeline can be run at a larger scale, across several files at once in a shorter period of time. The decision-making of the workflow is controlled by a configuration file that can be provided by the user (see example below, Fig.3.1), or using the default internal settings.

The workflow for stage 1 comprises of 3 modules performing a specific step: *runfastqc*, *checkfastqc* and *runtrim*. The workflow as a whole executes on a set of paired-read genome sequence files, with varying levels of parallelism via multithreading. The first step, *runfastqc*, can be executed on several files at once, with multithreaded execution at a ratio of 1 file per CPU thread. This step produces quality summary files, checking a series of metrics and producing a Pass/Warn/Fail flag for each metric. Step 2, *checkfastqc*, is a serial module, which decides whether to perform trimming, re-run *fastqc* on the trimmed files, and also whether the files are suitable for further analysis based on the configuration provided and the output of *fastqc*. Trimming the files is, unsurprisingly, managed by the *runtrim* step, which uses the tool ‘*cutadapt*’. This tool can make use of multithreading to improve performance. The main workflow execution is managed by a single executable script which performs a series of useful tasks, such as checking files are in correct locations, parsing command line options, setting the correct internal variables, and executing each of the stages as required.

```

# Names of each section should match FastQC summary.txt output. For each
# section declare how the variables change qcpass, qtrim (quality = 20
# trim), atrim (adapter sequence trimming) and recheck (re-run through FastQC
# after trimming). Can also set these under 'pass1/2', which changes behaviour
# depending on whether the files have already been trimmed. Pass1 = original
# files, Pass2 = files post trimming. Currently only set up to cope with one
# recheck loop. May be able to do more in later versions.

Basic Statistics:
  WARN:
    qcpass: False
  FAIL:
    qcpass: False

Per base sequence quality:
  pass1:
    WARN:
      qtrim: True
    FAIL:
      qtrim: True

Per tile sequence quality:
  pass1:
    WARN:
      qtrim: True
    FAIL:
      qtrim: True

Per sequence quality scores:
  WARN:
    qcpass: False
  FAIL:
    qcpass: False

Per base sequence content:
  pass1:
    FAIL:
      qtrim: True
      recheck: True
  pass2:
    FAIL:
      qcpass: False

```

Fig. 3.1 – Example YAML configuration file for the checkFastQC step of the workflow

- *Alignment*

The second workflow stage controls the alignment of a pair of genome sequencer reads. As with before, each main step consists of separate modules and a standard command line tool which runs the workflow defined in previous deliverables. The initial workflow provided by our partners contained a first stage that was comprised of several tools (*BWA Mem*, *Sambalster*, *Samtools*) connected via pipes and redirects. In total, up to 12 instances of these tools were in use at any one time, some of which supported multithreading. From previous benchmarking, we find that manually assigning threads to each tool capable of multithreading provided no benefits (and in some cases were perhaps slower) that just assigning the maximum threads to all tools. This is because *BWA Mem* is the biggest use of resources in this stage. We presume that the system used therefore makes good use of the resources available. Further work on balancing threads between these tools could yield further improvements, but we focussed our time on ensuring all stages of the workflow were available for re-use, with the hope of returning to this at a later date. The biggest change from the partner

workflow came from the release of *GATK4.0* towards the later stage of the project. This release moved to an open-source model, and relied on a Spark-based implementation to provide single-node multithreading and multi-node scaling capabilities for tools, in our case *BaseRecalibrator* and *ApplyBQSR* (the latter of which replaces the *PrintReads* stage for *GATK3.x*). Benchmarks for *GATK3* show that above 20 threads, performance increase with additional threads levels off, as shown below (Fig.3.2). This could be due to a bottleneck in allocating resources internally, and managing the large amounts of data. *GATK3* was also limited to running on a shared-memory (single-node) environment. From early benchmarking with *GATK4.0*, from single-node multithreading to multi-node Spark clusters, we find better scaling across a higher number of threads (Fig.3.2). Interestingly, while the run time for BaseRecalibration increases for *GATK4* compared to *GATK3*, the overall run time is slower on Cirrus, for a single node.

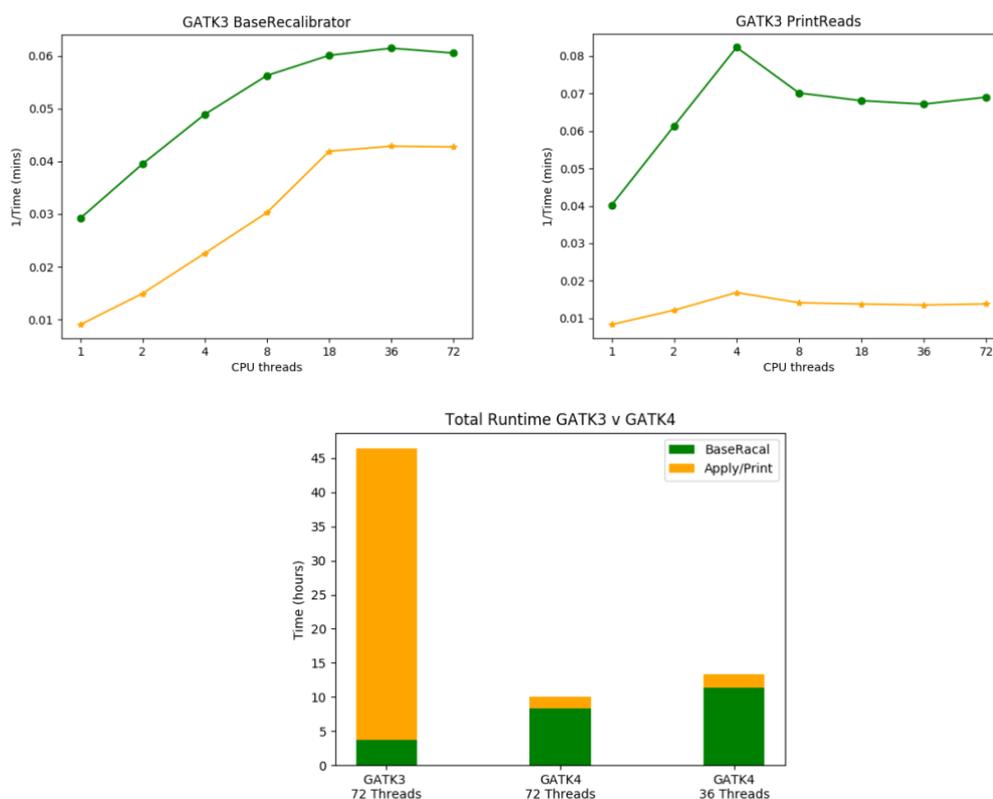


Fig. 3.2 – Top: Comparison of BaseRecalibrator and PrintReads for GATK3 showing poor scaling with thread count. Bottom: Comparison of total run time for GATK3 vs GATK4.

As with sequence quality control, each main step of the workflow will be self-contained in its own module, with a main command line tool, allowing users to run each stage individually, in a workflow manager, or imported into other Python scripts. These modules cover four main steps. The first steps are alignment and indexing of the read pairs, performed by *BWA Mem* and related tools. These output a single BAM file. The next step is Base Recalibration with *GATK* given the new alignment. Since some regions/bases of the DNA now contain more reads than others due to the alignment process overlapping regions, we need to re-calculate calibration statistics for the whole sequence.

Once this has finished, the final stage (*PrintReads* in *GATK3.x*, *ApplyBQSR* in *GATK4.0*) applies this recalibration to the aligned BAM file, resulting in a sorted, indexed, calibrated BAM file for further analysis.

- **Availability**

The Sequence Quality Control tool is available on the BioExcel GitHub, with the Alignment stage to follow soon. We are also aiming to include the packages within the *BioConda* package environment. This will help simplify not only the installation of our packages, but also the pre-requisite tools needed for each stage. An additional bonus is that each *BioConda* package has a corresponding *Docker BioContainer*, further increasing the usage of our tools, enabling ease of use with cloud infrastructure.

Sequence Quality Control is the most complete in terms of workflow, and will be available within days, if not already done so. Alignment will be soon to follow in the coming weeks. Documentation is also in the process of development, with basic usage and pre-requisites detailed.

- **Feedback**

Partner feedback during early development was very useful for understanding the needs and issues arising from their particular use case. However, as work progressed, our aims diverged. It was decided that the partners were unable to provide adequate time to complete all of the steps originally discussed in the first deliverable, so we focussed our efforts on the first two stages of the workflow. Towards the end of the project, in the process of verifying small details of the workflow, we were also informed that the tools being used by our partners in the alignment stage have changed. For the first step of the workflow, rather than using several pipes and redirects, the command now uses just two tools: *BWA Mem* as before, and *Bamsormadup*. This is likely to help with balancing resources, as both of these tools are multithreaded. A secondary advantage to this is that the latter tool also performs simple quality control and trimming as part of its execution, meaning the first stage is only used in rare cases.

We also had to make a decision regarding the use of *GATK4.0*. Currently this is included over *GATK3* in the final release, due to performance improvements. However, the method of implementation for Spark-cluster execution within the workflow, as currently developed, requires further investigation. It may be that end-users must manually invoke *GATK/Spark* for the final stages of the workflow for multi-node/spark cluster execution, but this will be finalised towards the end of development.

3.2 Pilot Use Case 2: Free Energy Simulations of Biomolecular Complexes

- **Use case summary**

Development of the pmx package in the scope of WP1 has advanced to the level allowing for a robust setup of alchemical calculations for ligand modifications. This progress is of high interest for both academic community as well as industry. In particular, the possibility to automate protocols for estimating free energy differences between drug-like molecules binding to proteins is important for lead identification and subsequent optimization. Currently, commercial software is prevalent in pharmaceutical companies, however, open source solutions are sought after as well.

A collaboration between the BioExcel CoE and Janssen Pharmaceutica has been started as a part of WP3 use case “Alchemical Free Energy Calculations in Biomolecules”. The aim of the WP3 use case was to probe applicability of the pmx based non-equilibrium free energy protocols in assessing protein-ligand binding free energy differences upon ligand modification. For a high throughput free energy estimation, a workflow was developed which constitutes the use case in WP2.

- **Workflows**

The workflow for the alchemical ligand modifications differs from the amino acid and nucleic acid mutations, as the modification libraries cannot be pre-computed in advance. Due to the vast chemical space available to the small organic molecules a different approach to generating hybrid structures and topologies was pursued.

The workflow used by pmx creates a unique mapping for every ligand pair for which free energy difference is to be calculated. The workflow follows two main routes: 1) 3D alignment and superpositioning; 2) topological maximum common substructure (MCS) identification. Since both procedures have their strengths and weaknesses, pmx builds mappings following both routes and selects a higher scoring atom mapping based on an internal scoring function. A number of knowledge-based rules are incorporated into the workflow to ensure that the topologies generated based on the acquired mapping properly describe both physical end states as well as the intermediate alchemical connection between them.

The created atom mapping is further used by pmx to build hybrid structure and topology to be used in molecular dynamics simulations. The subsequent calculations utilizing the generated structures/topologies may follow any free energy calculation protocol chosen by the user, e.g. equilibrium TI, FEP or non-equilibrium fast growth TI.

The technical details of the workflow will be described in the scientific publication which is currently in preparation.

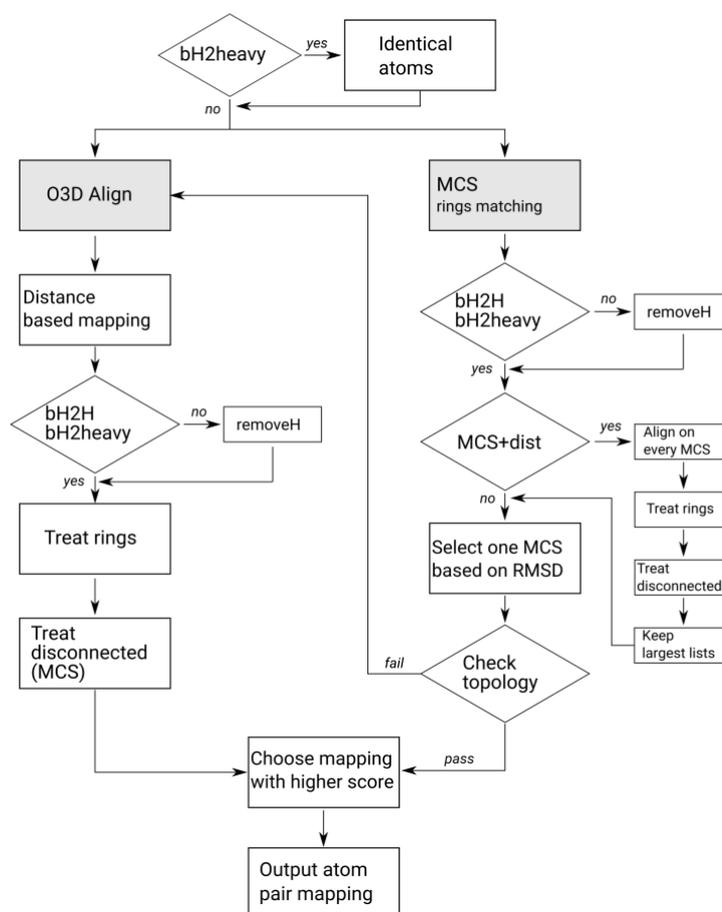


Fig. 3.2 – Atom mapping workflow for generating an alchemical morph between two ligands

- **Availability**

The pmx package is available online (<https://github.com/deGrootLab/pmx>) and is licensed under LGPL-3.0. The ligand modification modules are not yet included in the official master branch of the pmx repository and are available on request. Furthermore, amino acid and nucleotide mutations can be performed on the pmx webserver: <http://pmx.mpibpc.mpg.de/>

- **Feedback**

The ligand modification workflow is being tested in collaboration with Janssen Pharmaceutica. The main feedback comes from the industrial collaborators who apply the software in their work environment. The preliminary results of pmx application to the relative protein-ligand binding free energy calculations indicate that pmx outperforms or performs on par with the current state-of-art commercial software.

3.3 Pilot Use Case 3: Multi-scale Modeling of Molecular Basis for Odor and Taste

- **Use case summary**

Use case 3 allowed us to test on a real size and biological interesting problem the new and modern QM/MM interface developed within the BioExcel project that couples the quantum mechanical CPMD code with the molecular dynamics GROMACS program.

In particular, the biological system investigated in this use case is the enzyme adenylyl cyclase (AC) binding to the G-protein “Gas” and an ATP substrate. This is part of a wider project in collaboration with the Human Brain Project (HBP) that focusing on understanding the mechanism how this binding stimulates the synthesis of cyclic adenosine monophosphate (cAMP) from ATP substrate, amplifying signal transduction in the brain. A multi-scale approach, here a QM/MM one, is essential in this case because the synthesis of cAMP through the so called ATP cyclization involves a chemical reaction but the entire system is too large to be treated fully at quantum level. The systems AC bound to Gas (AC:Gas) and AC:Gas in complex with an ATP strand in the reactive conformation (AC:Gas:ATP) have been investigated by using both the original and popular QM/MM interface of CPMD, and the new and more efficient QM/MM interface.

Comparison of the simulations performed with the two interfaces allows us to assess the quality and reliability of the new interface and validate its use on state-of-the-art biological cases.

- **Workflows**

During the initial setup of the system, by using the original QM/MM interface, we had many difficulties to find a stable setup. This is due to inherent difficulties in modeling such a kind of systems. Performing a full trial and test approach in this case could be computational unaffordable due to the large computational costs of the quantum mechanical calculations. For this reason, we have developed a less computational demanding multistep approach that allowed us to find a suitable setup to perform the reference simulation with the original QM/MM interface. The pipeline of the proposed iterative multistep approach is the following:

1. Establishing of the quantities/properties that has be used to test the quality of the modeling
2. Initial modeling of the system
3. Equilibration step through traditional force field based level of theory
4. Intermediate QM/MM modeling by employing a lower QM level of theory (e.g. the computationally inexpensive semiempirical methods).
5. Test of this QM/MM modeling on the relevant identified properties: if the test fails, we go back to step 2.
6. QM/MM modeling by employing a more accurate QM level of theory (e.g. DFT method).

7. Test of this QM/MM modeling on the relevant identified properties: if the test fails, we go back to step 2.
8. Full QM/MM simulations at the higher level of theory.

Step 4 could be in principle divided in additional more intermediate steps, each one at an increased quantum level of theory. However, this was not necessary for Use Case 3 and therefore we do not explore this possibility.

- **Availability**

Building meaningful QM/MM models requires experience and mastering a significant large set of different programs. In addition, the workflow depicted in the previous section contributes to increase the complexity of the QM/MM investigation of relevant biological systems. For this reason, WP2 in collaboration with the team of the Use Case 3 have developed, implemented and deployed two virtual machines (VMs), available in the BioExcel Cloud Portal, to simplify the learning effort during training events for QM/MM approaches. Both VMs have been developed along with tutorials specifically designed for the modeling workflow related to the original QM/MM interface of CPMD based on GROMOS. However, a third VM, based on the new highly parallel QM/MM interface that couples CPMD with GROMACS is currently in preparation and will be available during the entire period of the BioExcel 2 project.

- **Feedback**

Since this Use Case is mainly a test case for the new QM/MM interface of CPMD, the current user feedback is limited to 1) the interaction with the HBP research group in getting a reliable model for the AC system, which brought us to develop the workflow described above, and 2) the first experiences with the tutorials and the VMs in the last training events, which have been used to improve some parts, in particular the ones involving the parameterization of the small ligands, initially performed through the usage of the Gaussian program and later replaced with alternative free software much simpler to be retrieved by a user.

3.4 Pilot Use Case 4: Biomolecular Recognition

- **Use case summary**

Interactomes are huge, intricate, and highly dynamic molecular networks that determine the fate of the cell. Our ability to understand the role of biomolecular interactions in both health and disease is tied to our knowledge of the atomic structures of both the interacting partners and their complexes. High-resolution experimental structure determination methods struggle with the high-throughput demand, calling for complementary integrative modelling approaches that combine automated workflows and cutting-edge software, leveraging the performance of HPC/HTC infrastructures supported by BioExcel.

The implementation of this use case requires combining HADDOCK, our in-house molecular docking software, with a molecular sampling method, typically molecular dynamics (MD) simulations performed using Gromacs, into a functional workflow.

- **Workflows**

Integration of GROMACS MD package with HADDOCK software has been possible using MDstudio. MDstudio is a microservice-based molecular dynamics framework, developed in the group of **Dr. Daan Geerke (molecular toxicology group, VU Amsterdam)**. MDstudio relies on crossbar.io, a Web Application Messenger Protocol (WAMP) that allows building distributed systems out of application components that are loosely coupled and can communicate in real-time (see <https://www.research-software.nl/software/mdstudio>). Through a collaborative partnership with **Dr. Daan Geerke group**, we integrated HADDOCK in MDstudio. In their current setting, both protein-small ligand docking and binding affinity prediction workflows are implemented in MDstudio, using both GROMACS and PLANTS (a renowned software for protein-ligand docking).

HADDOCK has been added to their initial design as a new component that allows to build Haddock .web parameter file and submit them to our server using a Haddock XML-RPC interface. The implementation relies on a graph-based data modelling library, something that resembles the Python graph library NetworkX, and an Object Relation Mapper (ORM). The library allows building arbitrary complex hierarchical data structures from sources such as .web files, JSON and JSON Schema files among others. In contrast to the actual HADDOCK data model, these graph models do not need to be compiled beforehand but are fully dynamic. It is a lean and hybrid data model that combines the behaviour of a classical Python object model with the power of modern databases. The ORM enables the application logic in the data model through pure Python classes. This allows attachment of class methods to pieces of data such as a PDB validator to haddock partners or an AIR builder to ambiguous restraint data.

The Haddock component can import and export .web files supporting all of the server functionalities. The data validation from HADDOCK has been replicated to

this new implementation and will raise an error if validation fails. The graph model will log a friendly warning and allow the user to fix the problem before submitting. This is important because it allows incremental construction of a new HADDOCK project by adding new partners, restraints, etc, when they become available in a workflow or when they are added by a user in a GUI. The workflow, managed through a workflow manager (which also uses the graph data model), is available but the GUI not yet.

- **Availability**

Available code related to this use case, corresponding to the MDstudio microservices, is accessible through the Netherlands eScience Center research software repository (<https://www.research-software.nl/software/mdstudio>).

- **Feedback**

None so far.

3.5 Pilot Use Case 5: Virtual Screening

- **Use case summary**

Virtual screening is a computational technique used in the early phases of a drug discovery in almost every pharmaceutical company nowadays. It is performed as the initial step in the selection of starting points suitable for further development. The computational cost involved usually precludes the consideration of receptor variability either due to sequence or conformational changes. A complete experiment including these aspects is beyond the computational capabilities of normal users. However, BioExcel software and practices can provide a bridge for these operations to large scale HPC, making possible to perform complete VS experiments in a competitive time scope. The collaboration with Nostrum Biodiscovery in the design process of this pilot use case has helped us to start a couple of relevant scientific studies that have actually driven the development and tuning of our software, and also have brought this effort to a real pharmaceutical discovery context.

- **Workflows**

A [complete workflow](#) has been designed, developed and tested in the BSC infrastructures, and is available in the BioExcel GitHub repository. It has been built using the BioExcel [building blocks](#), following the best practices for software development presented in the WP2 deliverables [D2.2](#) and [D2.3](#). This ensures a level of flexibility that is of a crucial importance in this kind of biomolecular workflows: the building blocks included in the pipeline, wrapping particular tools, are designed to be easily replaced, removed, and connected to new building blocks, thanks to the implemented interoperability. As an example for this particular case, the docking method ([Autodock Vina](#)) could be easily replaced for another command line software (e.g. [DOCK](#)).

The pipeline was tested with a couple of particular examples of great interest in the pharmaceutical industry, the Pyruvate Kinase and the Epidermal Growth Factor Receptor (EGFR) proteins, both already presented in a previous section of this deliverable and also in D2.3, both being good candidates to be used in a validation project, because of their whole body of knowledge available from the different scientific and pharmaceutical studies (EGFR is a real target that is nowadays exploited in the clinics) that can be used for setting up and fine-tuning the workflow.

The first test was run in the BSC premises, using the Marenstrum IV supercomputer and the Pyruvate Kinase protein. In this case, an ensemble of 4 structures coming from a 10ns-length Molecular Dynamics simulations were used, just as a proof of concept. The ensemble was automatically generated using the clustering tool offered by GROMACS package from the generated trajectory. [Autodock Vina](#) software was run from these target structures using a set of 20 active molecules extracted from [DrugBank](#) database, together with a number of 30 decoys downloaded from the [DUDE](#) database. This first test allowed us to find different issues and performance problems that were successfully solved. Now

the efforts have been put onto using the extensive ensemble of target structures generated by the EGFR mutations study (see section 2.5) with the VS pipeline.

- **Availability**

The pipeline used in the use case is available from the BioExcel GitHub repository:

<https://github.com/bioexcel/virtualsecreening/blob/master/workflows/vs.py>.

As it is composed of building blocks from the BioExcel software library (biobb), it can be easily installed and run following the steps that are explained in the GitHub repository home page: <https://github.com/bioexcel/virtualsecreening>.

New versions of the pipeline, adding the new functionalities implemented in the biobb modules, will be uploaded to the same repository.

- **Feedback**

The collaboration with Nostrum Biodiscovery in this development process was pivotal. A pharmaceutical point of view, rather than a technical one, is clearly needed in this particular use case. They have guided us to what is important for them, given suggestions about how to improve it, and proposed the scientific use cases. One important aspect that should not be neglected is that a complex scientific use case is usually not feasible during the lifetime of a project like this, it requires time to design, implement, test, run analyses, and in most of the cases, all of this points iteratively. Thankfully, Nostrum Biodiscovery will be a new partner in the BioExcel-2 project, which will ensure the continuation of this VS use case, together with new proposed ones.

4 Deployment Options

The plan designed in the early stages of the project of having two different deployment environments, one at BSC, to be used for verification and testing, and another one at EMBL-EBI, as a production one, accessible from the BioExcel Portal, was revealed to suit very well the needs of the BioExcel workflows development process. The current state of these infrastructures is detailed in the following sections.

4.1 Development Cloud Infrastructure (BSC)

4.1.1 Overview

BSC testbed infrastructure is being used for the whole development process of all the BioExcel building blocks and workflows built up from them. The development cloud infrastructure offers a virtualized system through an *OpenNebula* private cloud, which is able to dynamically instantiate new Virtual Machines. These VMs are used to test the software library: installation, configuration, and pipeline executions. Contextualization processes installed on these machines allow the almost-transparent transfer to different cloud environments (e.g. *OpenStack*).

BSC is a privileged environment for a testbed. A part from the cloud infrastructure, it offers direct access to a set of highly powered supercomputers, with usual architectures (Marenostrum IV, CPU-based) and also new ones (Minotauro, GPU-based or CTE-POWER, with IBM Power9 processors).

The COMPSs programming model, directly available in all these machines and cloud infrastructure, provides direct implicit parallelism to the pipelines executed, while at the same time is able to control the virtualization layer, making it transparent to the user, allowing to execute the same workflow in a series of environments, from single workstations, to HPC or grid/cloud facilities.

4.1.2 Available Tools

The whole set of BioExcel building blocks has been developed in the BSC premises, and are available from all the different machines installed in the center: Marenostrum IV, CTE-POWER, Minotauro and OpenNebula VMs. Workflows built up using these building blocks are also available on the same machines: *Model Protein Mutants* and Virtual Screening (presented in this document), and a set of useful pipelines for molecular dynamics simulations: protein energy refinement, GROMACS MD setup and ensemble generation. A local installation of Galaxy platform is running in an OpenNebula VM, with the biobb software library installed and being tested (publicly available early 2019). Another pair of VMs, shared with the Multiscale Complex Genomics (MuG) project, were developed and successfully cloned in the production infrastructure

(see section 4.2): Chromatin Dynamics and NAFlex. Access to any of these resources can be offered under BSC center permission.

4.1.3 Use Cases

- Virtual Machines developed and tested in the BSC cloud infrastructure were cloned in the EMBL-EBI cloud infrastructure and used in one training event ([Joint MuG-BioExcel workshop on Multi-resolution Nucleic Acids simulations](#)).
- PyMDSetup, a VM with the *Model Protein Mutants* workflow and all its software dependencies installed was developed and tested in the BSC cloud infrastructure and cloned in the EMBL-EBI cloud infrastructure and presented in the [BioExcel Community Forum](#). The same VM was registered and uploaded into the EGI AppDB, and used in serial and parallel executions managed with COMPSs programming model. COMPSs was able to dynamically deploy 2 VMs using EGI Federated Cloud, to compute the Molecular Dynamics simulations of 2 protein variants using the workflow (see benchmark in section 2.2.3).
- The *Model Protein Mutants* workflow was installed in the Marenstrum IV supercomputer and used together with PyCOMPSs (the Python binding of COMPSs programming model) for a couple of technical and scientific challenges, presented in this document (sections 2.4 and 2.5).

4.2 Production Cloud Infrastructure: BioExcel Cloud Portal (EMBL-EBI)

4.2.1 Overview

The BioExcel Cloud Portal has been developed at EMBL-EBI to simplify the experience for researchers when running their workloads across different cloud providers. The portal enables cloud providers and research teams to package and share their BioExcel applications and environments with their respective users. That way, these key applications can be deployed on demand onto the cloud providers the user is able to access.

From the BioExcel web portal a user is able to select the BioExcel resource directly (if it is offered online as a service), find the HPC centres where it is already installed and available for use and how they can access the software, or to retrieve the software from a repository (currently planned to be the EGI Applications Database but other solutions may be supported) and deploy the virtual machine or container through the EBI Cloud Portal API onto a cloud provider.

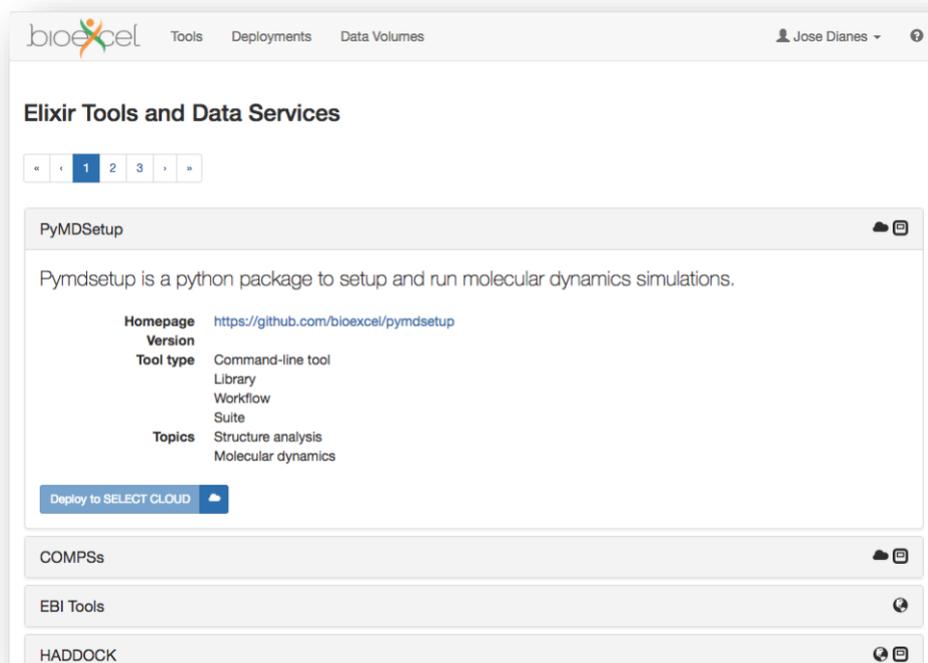


Fig. 4.1 – The BioExcel Cloud portal User Interface linking ELIXIR bio.tools (<https://bioexcel.ebi.ac.uk>)

The overall architectural design of the Cloud Portal has advanced considerably over the last two years. The REST API for programmatic access that serves the BioExcel is now capable of dealing with Teams or cloud communities as a way of sharing compute, data, and cloud resources. This is now a central concept for the BioExcel Cloud Portal that tries to satisfy the need for collaboration and reproducibility that we have detected while doing user research within the BioExcel user communities. This means that there is an asymmetric relationship between the owner of cloud resources (e.g. project lead with access to budget), those with the expert knowledge about cloud development (e.g. BioExcel Cloud Portal application developers), and the end user that tries to run computations to answer scientific questions. The portal proposes a model that allows the secure and efficient sharing of these three elements (resources as Configurations, skills as Applications specific to solve scientific use cases) in order to collaborate towards a better science.

Considering this model, it is important for those who own and share cloud resources to be able to track its usage. In order to do so, the BioExcel Cloud Portal API integrates with ELK (Elasticsearch and Kibana in our case) in order to keep track of cloud deployments and monitor them visually.

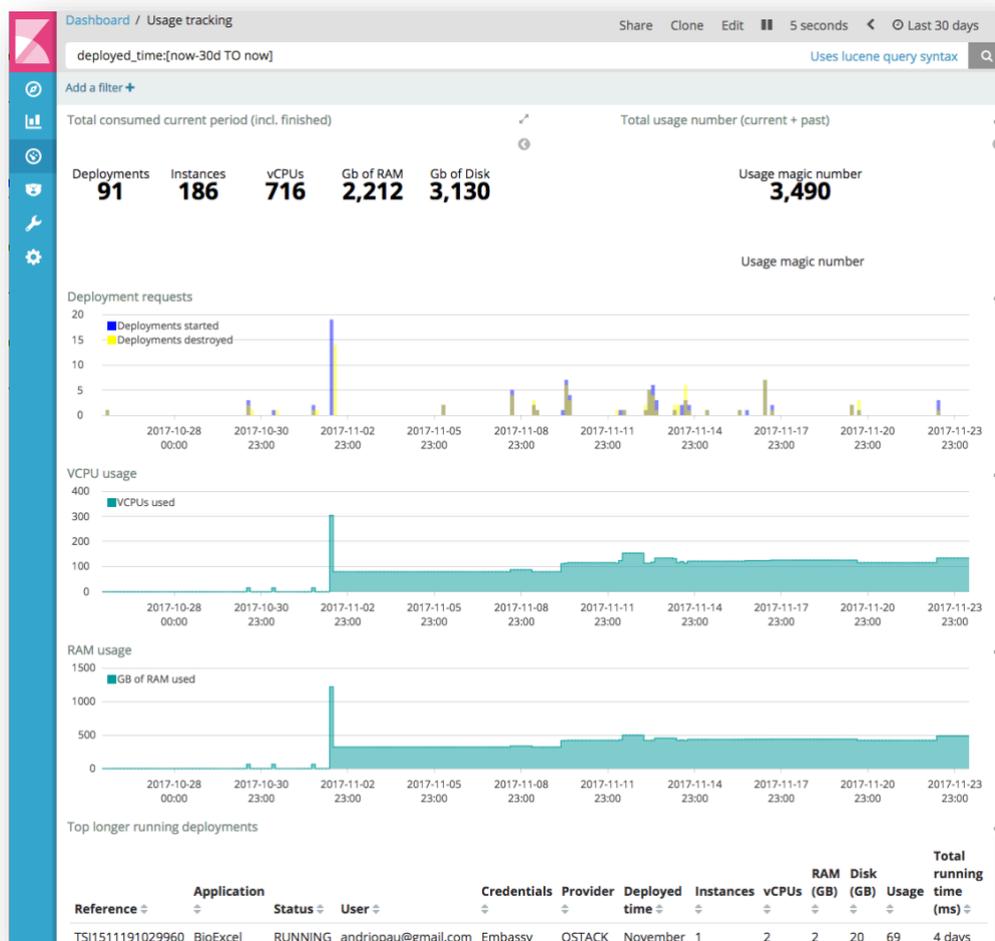


Fig. 4.2 – Analytics and usage tracking for the BioExcel Cloud Portal

The BioExcel Cloud Portal makes good use of two ELIXIR resources. The first is the ELIXIR Authentication and Authorisation Infrastructure (AAI), which is used as an identity provider, within the Authentication, Authorisation, and Profile service being developed EMBL-EBI in order to provide Single Sign On capabilities to the BioExcel Cloud Portal users. The second is the use of the ELIXIR application registry (bio.tools) in order to provide an application storefront to the BioExcel portal users. By doing so, BioExcel benefits from a well-known directory of biomedical applications, while directing users to the usage of the BioExcel Cloud Portal within the BioExcel project. EMBL-EBI has been collaborating closely with the bio.tools development team in order to bring new features and data representations to better represent the kind of applications and users that the BioExcel project will serve.

4.2.2 Available Tools

The list of available tools is public through both, bio.tools and the BioExcel Cloud Portal tools repository (<https://bioexcel.ebi.ac.uk/biotools>). There are different types of tools available, not all of them ready to be deployed

in a cloud resource. Those that are cloud ready, allow authenticated users to deploy them using their own or shared cloud resources.

The following tools are currently available as cloud-ready virtual machines:

- NAFlex: a VM to explore Nucleic Acids flexibility through a wide range of analyses performed on a trajectory produced by a Molecular Dynamics simulation.
- PyMDSetup: a VM including the PyMDSetup Python package to setup systems to run molecular dynamics simulations.
- Chromatin Dynamics: a VM with all necessary tools to create coarse-grained, 'beads-on-string' like representations of a chromatin fiber from just a DNA sequence (linker), and the nucleosome positions, and from them, generate a set of possible chromatin structure conformations.

All of them are provided as OpenStack-compatible VMs, and available at the EMBL-EBI Embassy Cloud to be deployed on demand.

4.2.3 Use Cases

The BioExcel Cloud Portal has proven to be a tool that facilitates the reproducibility of some of the BioExcel use cases, specially during training and workshops. There are three parts that need to be available during one of these workshops:

- The *applications*, or cloud-ready virtual infrastructure that contains the compute. This refers to the virtual machines that will be deployed in the cloud provider. It typically contains the tools and environment that are needed during the workshop.
- A BioExcel Cloud Portal *team* with shared cloud credentials. The usual case for workshops is that attendants will deploy the applications in cloud resources given by the workshop organisation. This is done through the creation of a team where both, the applications and the cloud resources are shared, and where workshop attendants are added.
- A way of *sharing datasets* and results. Applications are deployed in isolation. They need to have access to datasets, and it is also beneficial to have a location where data can be written for future sessions. We have done this by providing an NFS server where applications are automatically connected to as clients.

5 Conclusions & Future Work

BioExcel portable environments for computing and data resources work package has been working for three years in the setup of biomolecular workflows and modular tools, easily deployable and operational in a wide range of computational infrastructures. This deliverable presents the work done, exposes the power of our software library approach, and demonstrate how BioExcel has paved the way for the success in BioExcel-2 period.

The software development process chosen for the generation of the BioExcel building blocks and their accompanying workflows has followed the recommendations of ELIXIR project, pushing towards interoperability and reproducibility. Accordingly, FAIR principles have been applied to the modular software, strengthening the link between BioExcel and ELIXIR. The result of this practice is a set of interoperable units based on a collection of Python wrappers encapsulating software components. The programming model provides an imperative feature for the project, its capacity to be employed in different workflow managers and executed in different infrastructures. Specification of the building blocks and workflows with Common Workflow Language (CWL) provides another level of portability and reproducibility. BioConda packages, environment modules, Galaxy tools or KNIME nodes prepared ease the installation and use of the library in different infrastructures. Pipelines produced by the project are accessible from the BioExcel Cloud Portal, where Virtual machines with implementations of the workflows and all the software dependencies already installed can be directly deployed.

The pilot use cases built up during BioExcel lifetime are good examples of the scientific outcome from developing and running biomolecular workflows. The current stage of the pipelines for each of the use cases has been presented, with references to their available code (GitHub) or implementation (VM). Feedback retrieved during the evolution of the use cases, vital for all of them, was presented in detail in D2.3, and has been summarized also here.

The recently approved **BioExcel-2** project, the continuation of BioExcel, represents a new opportunity to expand on the work done during the last 3 years in portable environments for biomolecular simulations. With the entire basis already set (software library, development and production interfaces), and the ball already rolling, the new proposed use cases are complex and challenging: biomolecular interactions (antibody design, interactome), drug design (with HPDA and machine learning) and electronic interactions (with large QM/MM systems). The *biobb* software library, which has already reached the production phase, will be updated with new and rich functionalities, going towards the state-of-the-art methodologies such as biased MD and data analytics, with at the same time retaining usability, interoperability and reproducibility. A key focus will be put in the possible convergence of High Performance Computing (HPC) with High-Performance Data Analytics (HPDA), where complex workflows involving HPC computations and HPDA operations will be combined. Finally, and following the work already started by the technical challenge presented in this deliverable,

new workflows and their corresponding workflow managers will be tested and optimized for the coming Exascale computing.

BioExcel-2 will also prioritize the external awareness and usage of our workflow building blocks approach to a wider audience. There are a variety of projects around the world developing workflow systems for biomolecular research, but to date it has been difficult to achieve widespread community uptake, or to develop a good model for sustainability. BioExcel wants to foster collaboration with all these projects (some European, some from the US), to work together on joining efforts.

A kick-off of this collaboration is already in preparation for a joint BioExcel-organized workshop together with US partners [MolSSI](#) and UK-based collaborators from [CCPBioSim](#). Together we have established that many of the same concerns that initially held back the field of genomics in their uptake of workflows (scaling, stability, interoperability, usability), have emerged, perhaps to an even greater degree, as challenges in the field of biomolecular modelling and simulations. Combined with our links to ELIXIR, EOSC, HPC and code developers we want to further develop and share best practice for workflow design and use across our communities.

The joint workshop will be split in two parts: the first one, in Barcelona (December 2018), will bring together current projects from both sides of the Atlantic that involve development and application of biomolecular workflows, with an aim to foster a dialogue to develop interoperable/harmonised solutions that have the best chance of being long-term sustainable products, aiming at a worldwide community of molecular simulation scientists using them day-to-day and supporting their continued development.

The second part, to be hosted in the US thanks to our [MolSSI](#) colleagues (early 2019), is thought as a biomolecular workflow hackathon, where all the synergies identified in the first meeting will be put into practice by code developers working cooperatively in the same room. Close collaborations hopefully emerging from this event will be reinforced during the BioExcel-2 lifetime, favoring the visibility and strength of BioExcel-2 workflows, and contributing to the larger biomolecular simulations community.

6 References

1. Oinn, T., et al., *Taverna: a tool for the composition and enactment of bioinformatics workflows*. Bioinformatics, 2004. **20**.
2. Wolstencroft, K., et al., *The Taverna workflow suite: designing and executing workflows of Web Services on the desktop, web or in the cloud*. Nucleic Acids Res, 2013. **41**(Web Server issue): p. W557-61.
3. Afgan, E., et al., *The Galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2016 update*. Nucleic Acids Research, 2016. **44**(W1): p. W3-W10.
4. Berthold, M.R., et al., *KNIME: The Konstanz Information Miner*, in *Data Analysis, Machine Learning and Applications: Proceedings of the 31st Annual Conference of the Gesellschaft für Klassifikation e.V., Albert-Ludwigs-Universität Freiburg, March 7–9, 2007*, C. Preisach, et al., Editors. 2008, Springer Berlin Heidelberg: Berlin, Heidelberg. p. 319-326.
5. Warr, W.A., *Scientific workflow systems: Pipeline Pilot and KNIME*. Journal of Computer-Aided Molecular Design, 2012. **26**(7): p. 801-804.
6. Goble, C.A., et al., *myExperiment: a repository and social network for the sharing of bioinformatics workflows*. Nucleic Acids Research, 2010. **38**(suppl_2): p. W677-W682.
7. Bhagat, J., et al., *BioCatalogue: a universal catalogue of web services for the life sciences*. Nucleic Acids Research, 2010. **38**(suppl_2): p. W689-W694.
8. Wilkinson, M.D. and M. Links, *BioMOBY: an open source biological web services proposal*. Brief Bioinform, 2002. **3**.
9. Wilkinson, M.D., et al., *The FAIR Guiding Principles for scientific data management and stewardship*. 2016. **3**: p. 160018.
10. 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 Suhecki, R and Svobodová Va?eková, 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 [version 1; referees: 3 approved]*. F1000Research, 2017. **6**(876).
11. Abraham, M.J., et al., *GROMACS: High performance molecular simulations through multi-level parallelism from laptops to supercomputers*. SoftwareX, 2015. **1–2**: p. 19-25.
12. Tejedor, E., et al., *PyCOMPSs: Parallel computational workflows in Python*. International Journal of High Performance Computing Applications, 2015.
13. Ison, J., et al., *Tools and data services registry: a community effort to document bioinformatics resources*. Nucleic Acids Research, 2016. **44**(D1): p. D38-D47.

14. Capella-Gutierrez, S., et al., *Lessons Learned: Recommendations for Establishing Critical Periodic Scientific Benchmarking*. bioRxiv, 2017.
15. Ison, J., et al., *EDAM: an ontology of bioinformatics operations, types of data and identifiers, topics and formats*. *Bioinformatics*, 2013. **29**(10): p. 1325-32.
16. Gruening, B.a.S., O and Moreno, P and da Veiga Leprevost, F and Ménager, H and Søndergaard, D and Röst, H and Sachsenberg, T and O'Connor, B and Madeira, F and Dominguez Del Angel, V and Crusoe, MR and Varma, S and Blankenberg, D and Jimenez, RC and null, null and Perez-Riverol, Y, *Recommendations for the packaging and containerizing of bioinformatics software [version 1; referees: 2 approved with reservations]*. *F1000Research*, 2018. **7**(742).
17. Blankenberg, D., et al., *Dissemination of scientific software with Galaxy ToolShed*. *Genome Biology*, 2014. **15**(2): p. 403.
18. Balasubramanian, V., et al., *ExtASY: Scalable and Flexible Coupling of MD Simulations and Advanced Sampling Techniques*. Vol. abs/1606.00093\, 2016.
19. Peter, A., et al., *Common Workflow Language, v1.0*. 2016, Figshare.
20. Soiland-Reyes, S., M. Gamble, and R. Haines, *Research Object Bundle 1.0*. 2014.
21. Zanella, A., et al., *Red cell pyruvate kinase deficiency: molecular and clinical aspects*. *British Journal of Haematology*, 2005. **130**(1): p. 11-25.
22. Valentini, G., et al., *Structure and Function of Human Erythrocyte Pyruvate Kinase: MOLECULAR BASIS OF NONSPHEROCYTIC HEMOLYTIC ANEMIA*. *Journal of Biological Chemistry*, 2002. **277**(26): p. 23807-23814.
23. Larsson, P., B. Hess, and E. Lindahl, *Algorithm improvements for molecular dynamics simulations*. *Wiley Interdisciplinary Reviews: Computational Molecular Science*, 2011. **1**(1): p. 93-108.

Appendix

Table A.1: BioExcel software library availability. PUC acronym refers to Pilot Use Case.

Module	GitHub	ReadTheDocs	BioConda	Pyp	BioContainer	bio.tools
Building Blocks						
biobb_*						
biobb_common				PIP		
biobb_io				PIP		
biobb_md				PIP		
biobb_model				PIP		
Workflows						
pymdsetup						
PUC1 – Cancer Genome Sequencing Data						
PUC2 – Free Energy Simulations (pmx)						
PUC3 – Multi-scale Modeling of Molecular Basis for Odor and Taste						
PUC4 – Biomolecular Recognition						
PUC5 – Virtual Screening						
Naflex						
Chromatin Dynamics						