

Project acronym: EOSC4CANCER
Grant Agreement Number: 101058427
Project full title: A European-wide foundation to
accelerate Data-driven Cancer Research
Call identifier: HORIZON-INFRA-2021-EOSC-01

D3.2 Integrative analysis of multiple data types within cancer portals

Version:	1.0
Status:	Final
Dissemination Level:	Public
Deliverable Type:	DEM
Due date of deliverable:	30.11.2024
Actual submission date:	29.11.2024
Work Package:	WP 3 Federated cancer data analysis through portals across Europe
Lead partner for this deliverable:	UiO
Partner(s) contributing:	UBx, Lygature

Abstract

This report details the advancements made in Task 3.2, focusing on the development of functionality for integrated analysis of cancer data for different standardised data types, including clinical data, radiomic images, time series data and DNA variant allelic frequency data, all in a portable format for use in federated analysis settings across different sites hosting relevant cancer data. It describes the integration of visualisation of radiomic images. Further, linking of the existing software called cBioPortal, which is used for analysis and visualisation of molecular data, to trigger analyses in predefined workflows outside of cBioPortal is presented. It links to the workflow engine called Galaxy, with automatic delivery of results back to cBioPortal of the analysis performed. This is realised using time series data on cancer mutations in a given tumor, based on sequencing at multiple time-points, followed by analysis of the existence of clones of cell in the tumor, through an open source third-party software, PyClone VI, and the provision of results back to cBioPortal for visualization in an integrated presentation. Workflows are integrated into portable systems using container technology for easy deployment for local operation in Trusted Research Environments, along with documentation and SOPs, and software and documentation, SOPs and guidelines are available through reference installations.

Main author(s):

Jeanne Chèneby	UiO
Eivind Hovig	UiO

Other author(s):

Aurélien Barré	UBx	Benjamin Dartigues	UBx
Robin Navest	LYG	Macha Nikolski	CNRS
Sophie Huiskes-Berends	LYG		

Revision History

Version	Date	Changes made	Author(s)
0.1	11.25.2024	Accepted contributions for missing parts and made minor edits for readability.	Eivind Hovig (UiO)
0.2	11.27.2024	Updated missing elements, and made edits through reviewer comments.	Eivind Hovig (UiO)
0.3	28.11.2024	Refinement of Abstract and Executive summary, and minor text edits.	Eivind Hovig (UiO)
1.0	28.11.2024	Edits accepted on workshop report. Report finalized.	Eivind Hovig (UiO)

Statement of originality

This deliverable contains original unpublished work except where clearly indicated otherwise. Acknowledgement of previously published material and of the work of others has been made through appropriate citation, quotation or both.

TABLE OF CONTENTS

EXECUTIVE SUMMARY	6
1 Introduction	7
2 Image integration in cBioPortal	9
3 Reference installation of cBioPortal	11
4 cBioPortal integration with Galaxy for analysis of novel data types	12
Key Features	12
4.1 User workflow	12
4.1.1 Pre-requisites	12
4.1.2 Exporting Data from cBioPortal	13
Using the export button	13
Data Formats	14
4.1.3 Analysing data on Galaxy	14
4.1.4 Exporting data from Galaxy to cBioPortal	16
4.2 Live demonstration	17
4.2.1 Access and registration	17
4.2.2 Security Considerations	17
4.3 Methods	18
4.3.1 Current limitations	18
4.3.2 Proposed Solution	18
4.3.3 Data Extraction from cBioPortal	18
Overview	18
API Usage	18
Data Export Functionality	18
4.3.4 Intermediary Server	19
Overview	19
Functionalities	19
Export Data from cBioPortal to Galaxy	20
Export Data from Galaxy to cBioPortal	20
4.3.5 Galaxy integration	22
Overview	22
Challenges in Automating Workflows	22
Created Tools	23
4.3.6 Code Availability and Deployment Instructions	25
GitHub Repositories	25
Docker and GitHub Actions	26
Running the Full Project	26
Individual Component Documentation	28
4.3.7 Security Considerations	28
4.4 Discussion	29
4.4.1 Key benefits	29
4.4.2 Future considerations	29
5 Conclusion	29

LIST OF FIGURES

Figure 1. Screenshot of radiology image integration through XNAT	9
Figure 2. Screenshot of WSI integration through xOpatT	10
Figure 3. Simplified system architecture of the cBioPortal-Galaxy integration	12
Figure 4. cBioPortal patient page with Galaxy data export button	13
Figure 5. Show a prompt to collect necessary information from the user for exporting to Galaxy	14
Figure 6. Data exported from cBioPortal with name data_studyID_casID	14
Figure 7. New Galaxy tools	15

EXECUTIVE SUMMARY

This document describes the development of functionality for integrated analysis of cancer data for different standardised data types, including clinical data, radiomic images, time series data and DNA variant allelic frequency data, all in a portable format for use in federated analysis settings across different sites hosting relevant cancer data. The portability of the integration is crucial to enable federation. The federation aspect caters to the handling requirements of the human sensitive data to comply with FAIR data practices. The implementation utilises well established cancer analytical solutions, and provides a bootstrap for further development and local adaptation. Examples are provided, along with tutorials and guides, to facilitate dissemination. One main feature that has been developed is that of linking the existing software called cBioPortal, which is used for analysis and visualisation of molecular data, to trigger analyses in predefined workflows outside of cBioPortal, in this case to the workflow engine called Galaxy, with automatic delivery of results back to cBioPortal of the analysis performed. This is realised using time series data on cancer mutations in a given tumor, based on sequencing at multiple time-points, followed by analysis of the existence of clones of cell in the tumor, through an open source third-party software, PyClone VI, and the provision of results back to cBioPortal for visualization in an integrated presentation.

1 Introduction

Cancer is fundamentally a genetic disease, in that genetic alterations causes cellular deregulation, leading to uncontrolled growth and metastasis formation. The genetic alterations are partly predisposed through heritability, and partly occur as consequences of stochastic and environmental. Given the large number of cancer types, and that they vary in their genetic constitution and exist in different normal genetic backgrounds, tend to make each tumour unique. The task of understanding the defining features for diagnosis and treatment of each tumour requires collections of large numbers of cancer cases, in order to find the systematic features that are targetable, in the context of personalised medicine. Given the nature of many of the core molecular data as being defined as sensitive in a GDPR context, and given the many types of data that are relevant in the analyses of such data, coupled with the size of the data, activates the need to enable integrative analysis across data being hosted in different systems, e.g. Trusted Research Environments (TREs). To enable this, given that the handling of sensitive data essentially means to restrict access for security reasons, it is necessary to facilitate integrative approaches, and have these operating across sites. Further, from a researcher perspective, the accessibility of suitable tools and the ease of use of analytical approaches should be high, to reap the potential residing in European health data. This also requires the use of standardised data types in order to enable joint analytical steps on data from different sites.

A large number of data types exist in cancer research, and the data resides mostly in disparate sites, while the data are increasingly being generated in the context of the TRE environments in the member states, currently subject to a combination of national and EU regulations. A large amount of analytical tools exists in cancer research, in the form of tools developed for dedicated fine-grained analytical purposes, or in the form of larger workflow engines, where multiple analytical steps are analysed serially or in parallel, and further in the form of integrative analyses of multiple data sets.

In molecular cancer research, with the advent of low cost sequencing, it is possible to accrue large sets of sequenced samples. The analysis of these data sets often requires large numbers of steps, where each step needs documentation and can often be performed in many different ways. The understanding of the data will also often rely on visualisation of the data in informative ways, often also to trigger the need for further analytical steps. The linking of analytical steps through the use of workflows can often be very efficient. Such workflow engines are several, among which are Nextflow and Snakemake that target bioinformaticians who are proficient in scripting languages, while Galaxy are more inclusive to end-users without programming skills (workflow development may occur through drag-and-drop actions on predefined steps).

One important approach to handle the data is that of federated analysis, meaning that the same analytical procedure is applied across different data sets from different locations. This may be achieved through the use of software container technology, where the software is made portable and relatively independent of the operating system that may exist in each environment. Container technology enables sharing of software in a way that facilitates a consistent analysis to be performed at each site, and then the summary data may be subsequently shared, as these data may be defined as non-sensitive.

A basic premise of EOSC4Cancer is that users will be able to benefit from the access to additional data types and methods in the visualisation and analysis portals they are familiar with in their research, and such that they can be combined with the data where they reside. Task 3.2 concentrates efforts on the cBioPortal, as a well-established solution to data handling, advanced visualisation and longitudinal views of patient characteristics before, during and after cancer treatment. The aim has been to enable efficient analysis processes

by both adhering to the FAIR principles and the best practices for research software, as well as maintaining separate software management for each component to enable close monitoring and assessment, and to better support integrative analysis of multi-type cancer data.

cBioPortal provides functionality for a number of settings, including detailed overviews over cancer patients and cohorts, with possibilities to drill down on molecular changes in a given gene, or in providing graphical and statistical information on e.g. molecular marker frequencies and a number of other aspects. In a sense, it can function as a digital pilot in exploration of cancer data sets, based on standard format inputs. In order to expand the features of cBioPortal for cross-European federated and transparent analysis of cancer-related data, cutting-edge analysis methods and protocols have been developed, as is described in the implementation reported below.

We demonstrate that is imaging data are now integrated into cBioPortal, and in this way enable the ability to jointly observe morphological characteristics of the tumor, combined with relevant clinical variables, in the form of a link to the OHIF viewer in XNAT toolkit designed for sharing and storing medical imaging data.

This deliverable also demonstrates how to enable new parts of a workflow utilizing third-party software, to execute this functionality on test and real data, and to subsequently visualise the results within cBioPortal. One important aspect of cancer is that every tumor is subject to evolutionary processes happening in the patient as the cancer develops over time, and also as a response to treatments. The evolution will generally result in clones developing within the tumor that may or may not succeed in growing more than the other cancer cell. Over time, some clones will prevail, some will disappear, and some new may arise, mostly as a result of the challenges posed by the tumor microenvironment and treatments. The time component to observe the clonal evolution is here implemented in the form of an analysis across several sequencing analyses of a given tumor at several time points in the evolutionary process. In this way, it may be possible to better understand which molecular features may be critical to target in the further treatment history of the patient, as well as to understand more of the dynamics of tumor evolution.

A workflow has now been developed that takes the tumor variant allelic frequencies of each variant into account, as derived from sequencing efforts, and subjects this through a cluster analysis using third-party open-source software (PyClone VI), which has been implemented as a workflow in Galaxy. It is now possible to select relevant patients with the relevant allelic frequency information across several sequencing events, trigger the analysis from within cBioPortal, export the data to Galaxy, followed by automatic execution of the clonality analysis, and where the results of the analysis in the form of data with graphs become linked to the data in cBioPortal, and are readily visualised there.

A number of reference installations have also been developed, and many features are now available in containerised form for local deployment in disparate research settings, and these may also be modified according to local needs, according to user manuals for best practice to guide their use.

2 Image integration in cBioPortal

For the image integration, we used the ability of cBioPortal to include other types of data through resource data¹. The linkage of an image viewer essentially entails adding resource data to the sample ID² or patient ID³ in the cBioPortal study. This resource can then be viewed in the embedded tab on the cBioPortal Patient View via iframe. This process is very flexible and we provide a demonstrator⁴ for two particular examples, in close collaboration with WP2 and WP4. In the first demonstrator for radiological images, a link to the OHIF viewer in XNAT⁵, the extensible neuroimaging archive toolkit, is an open-source solution designed for sharing and storing medical imaging data, especially radiology data, in DICOM format. In the second demonstrator, for histopathological images, a link to the xOpat whole slide image (WSI) viewer was established (see Figure 2). xOpat⁶, eXplainable Open Pathology Analysis Tool, is an open-source whole slide image viewer.

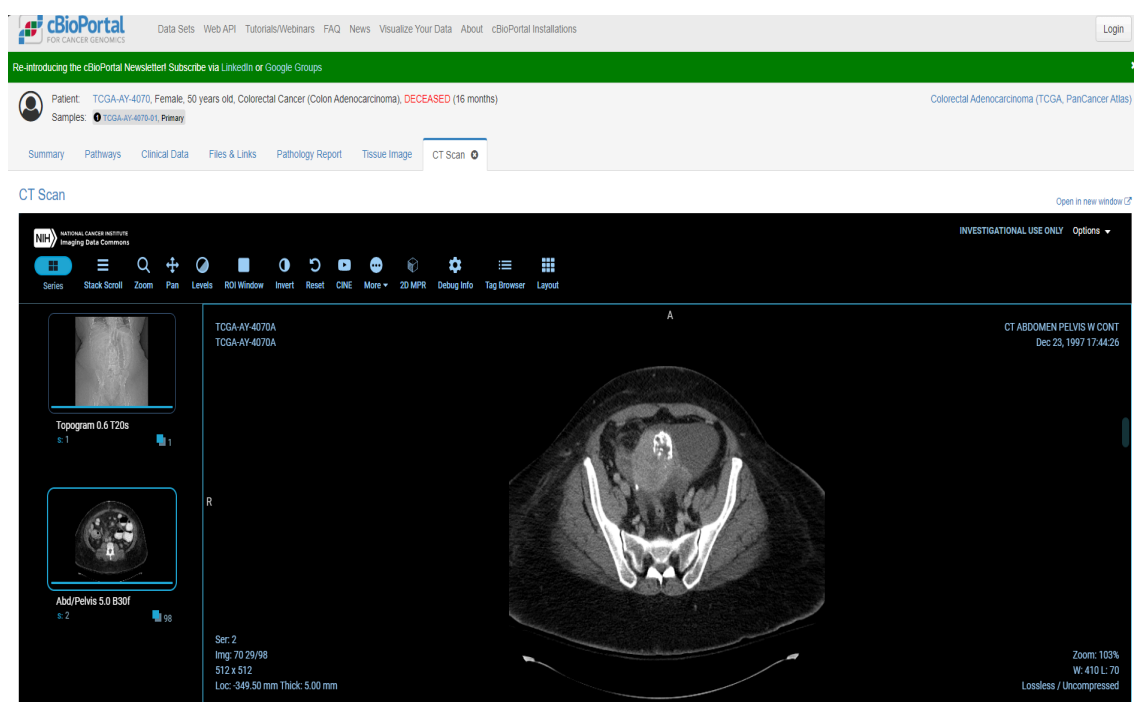


Figure 1. Screenshot of radiology image integration through XNAT

¹ <https://docs.cbioportal.org/file-formats/#resource-data>

² <https://docs.cbioportal.org/file-formats/#example-sample-resource-data-file>

³ <https://docs.cbioportal.org/file-formats/#example-patient-resource-data-file>

⁴ [EOSC4Cancer - Milestone Report - M6.pptx](#)

⁵ <https://link.springer.com/article/10.1385/NL:5:1:11>

⁶ <https://onlinelibrary.wiley.com/doi/10.1111/cgf.14812>

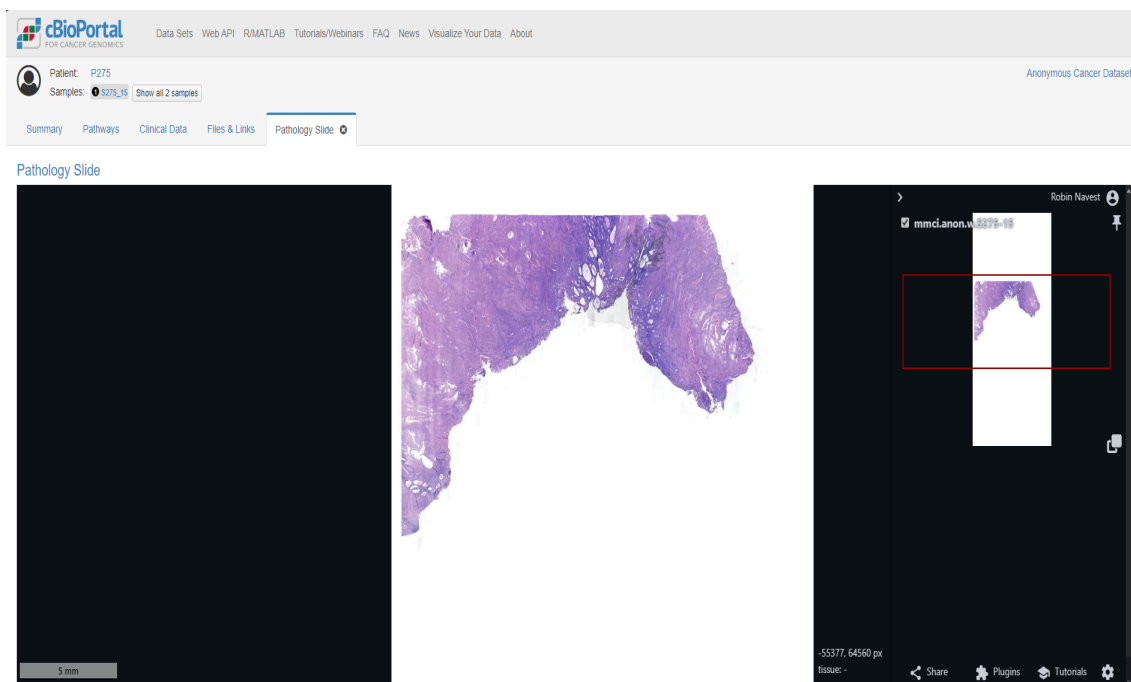


Figure 2. Screenshot of WSI integration through xOpat

It is envisioned that additional functionality besides the demonstrated image visualisation will be enabled. In combination with the integration of Galaxy as described in section 4, image analysis will be investigated in D2.2.

3 Reference installation of cBioPortal


Within EOSC4Cancer, there have been several efforts to develop integrative analysis of multiple data types for cBioPortal. Typically, these developments happened in different instances of cBioPortal, to have a clean setup:

- 1) The radiological image integration was developed on the EOSC4Cancer reference instances of cBioPortal and XNAT hosted by Health-RI
- 2) The histopathological image integration was developed on the cBioPortal and xOpat instances hosted by BBMRI-ERIC
- 3) The integrative analysis through Galaxy was developed on a dockerised instance of cBioPortal hosted by the University of Oslo

To enable the wider cBioPortal community to use all these developments, we collaborate with the cBioPortal development team at Memorial Sloan Kettering Cancer Center. Ideally, the modifications in cBioPortal required for the developed functionality will end up in the reference installation⁷ and should be properly documented. For reproducibility of the image integration, for example a manual is being developed⁸. It was agreed that this manual would form the basis for a section in the cBioPortal documentation⁹ on this topic, to ensure that the entire cBioPortal community would benefit from the developments created in EOSC4Cancer.

In addition, a cBioPortal workshop was organized early 2023. During this online workshop¹⁰, the users were provided an overview of functionalities in cBioPortal and practiced using exploring public study material. Data loaders, who needed to be familiar with curating and transforming study/genomic data, got hands-on experience with loading test data into a private cBioPortal with help from Health-RI cBioPortal operators.

⁷ <https://github.com/cbioportal/cbioportal/>

⁸  EOSC4Cancer_cBioPortal_image_integration_manual.docx

⁹ <https://docs.cbioportal.org/user-guide/overview/>

¹⁰ <https://zenodo.org/records/8163342>

4 cBioPortal integration with Galaxy for analysis of novel data types

This section presents a proof-of-concept designed to show the integration of Galaxy¹¹ with cBioPortal¹². The aim is to use mutation data from cBioPortal within Galaxy to **analyse the clonal population structure** for individual patients, and to get the results back into cBioPortal.

To do this, we have chosen PyClone-VI¹³, which is a tool for inference of clonal population structure from variant allele frequency (VAF) data as found in cancer genomics data, helping in understanding the genetic landscape of cancer evolution. Specifically, PyClone-VI is a reimplement of the original PyClone, leveraging Variational Inference (VI) for faster and more scalable computational performance. This tool is widely used for analysing intratumoral heterogeneity by identifying subpopulations of cancer cells (clones) and estimating their relative abundance.

The diagram below illustrates the system architecture for this project (Figure 3). Technical information can be found in the methods [section](#).

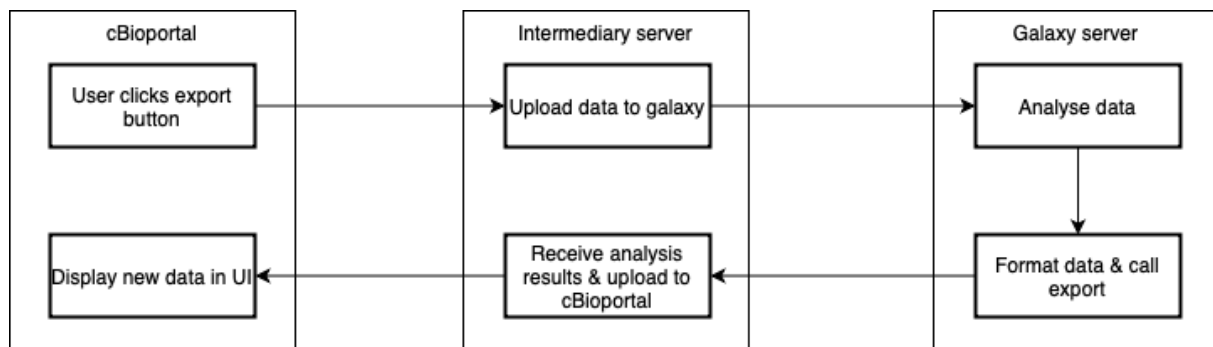


Figure 3. Simplified system architecture of the cBioPortal-Galaxy integration

Key Features

- ▶ An export feature for mutation tables in cBioPortal
- ▶ Data analysis using Galaxy
- ▶ Data return and SQL data upload to cBioPortal

4.1 User workflow

4.1.1 Pre-requisites

Users need to have access to a version of cBioPortal that is connected to Galaxy, with at least one study loaded with mutation data for multiple timepoints. The destination Galaxy

¹¹ Galaxy project: <https://doi.org/10.1093/nar/gkx410>.

¹² cBioPortal: <https://docs.cbioportal.org/about-us/>.

¹³ Gillis, S. and Roth, A., 2020. PyClone-VI: scalable inference of clonal population structures using whole genome data. *BMC bioinformatics*, 21, pp.1-16.

server is configured during the server setup by the administrator, making it unchangeable to users.

We are using PyClone-VI to infer clonal population structure. Refer to its documentation for more information on mandatory input data, as using patient data lacking the mandatory mutation information will result in a failure in the analysis¹⁴.

The Galaxy server needs to have the following tools:

- ▶ PyClone-VI
- ▶ Plot PyClone-VI output
- ▶ Export cBioPortal Image
- ▶ Export cBioPortal Timeline

Users also need an account on the Galaxy server with an API key.

4.1.2 Exporting Data from cBioPortal

Using the export button

For now, only cBioPortal web portal built from the following fork from the official front-end cBioPortal repo¹⁵ contain the "Export data to Galaxy" button (Figure 4).

266 Mutations (page 1 of 27)

Samples	Gene	Protein Change	Annotation	Mutation Type	Allele Freq	Cohort	COSMIC
1 2 3 4	BRAF	K601E	Missense	Missense	5%	94	
1 2 3 4	TP53	M237I	Missense	Missense	57%	149	
1 2 3 4	ARHGAP35	Y855*	Nonsense	Nonsense	6%		
1 2 3 4	KMT2C	Q2220*	Nonsense	Nonsense	6%		
1 2 3 4	CCNB1IP1	D232N	Missense	Missense	<1%		
1 2 3 4	CPS1	I794N	Missense	Missense	7%		
1 2 3 4	ATP2B3	X375_splice	Splice	Splice			
1 2 3 4	NTRK3	E100K	Missense	Missense	8%		
1 2 3 4	H2AC16	A41T	Missense	Missense	2%		
1 2 3 4	FLT3	R833L	Missense	Missense	3%		

Showing 1-10 of 266 Mutations

Figure 4. cBioPortal patient page with Galaxy data export button

Once clicked, the user must fill a prompt with their API key and the name of the history they want to export the data (Figure 5). If any error arises during the export, a modal will appear with a message to help understand what went wrong.

¹⁴ <https://github.com/Roth-Lab/pyclone-vi>

¹⁵ <https://github.com/elixir-oslo/cbioportal-frontend> (tmr branch)

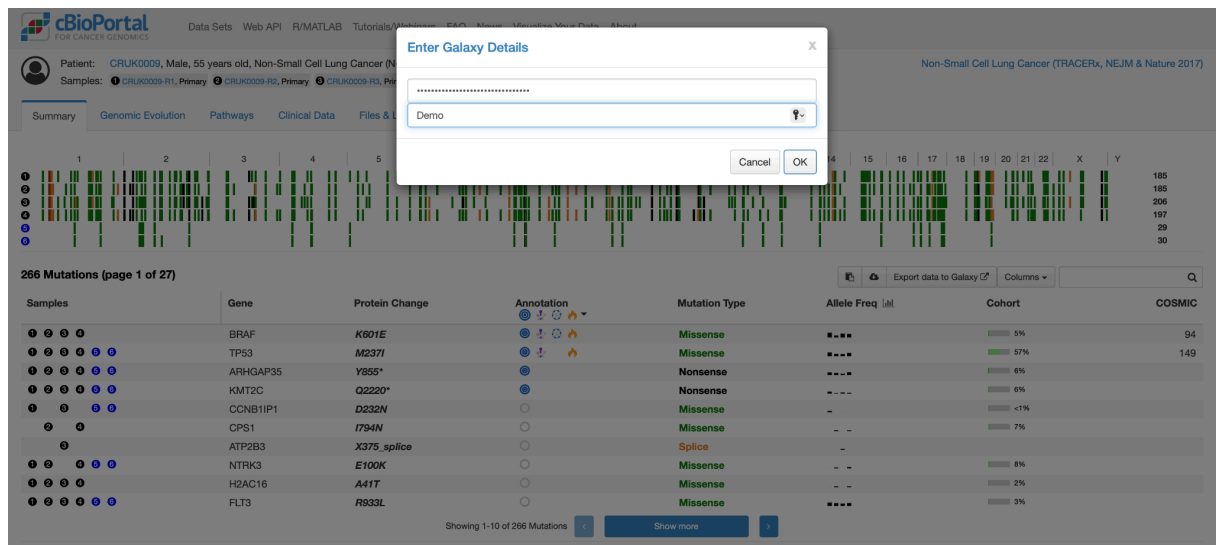


Figure 5. Show a prompt to collect necessary information from the user for exporting to Galaxy

Data Formats

- ▶ The complete raw mutation table for this patient, in tab-delimited format, will be transferred to Galaxy. Even with data filtering on the cBioPortal table, the export will include all data.
- ▶ Data will only be visible for the user on the specified history.

4.1.3 Analysing data on Galaxy

If no errors were encountered, data will be available at the specified user history in Galaxy (Figure 6).

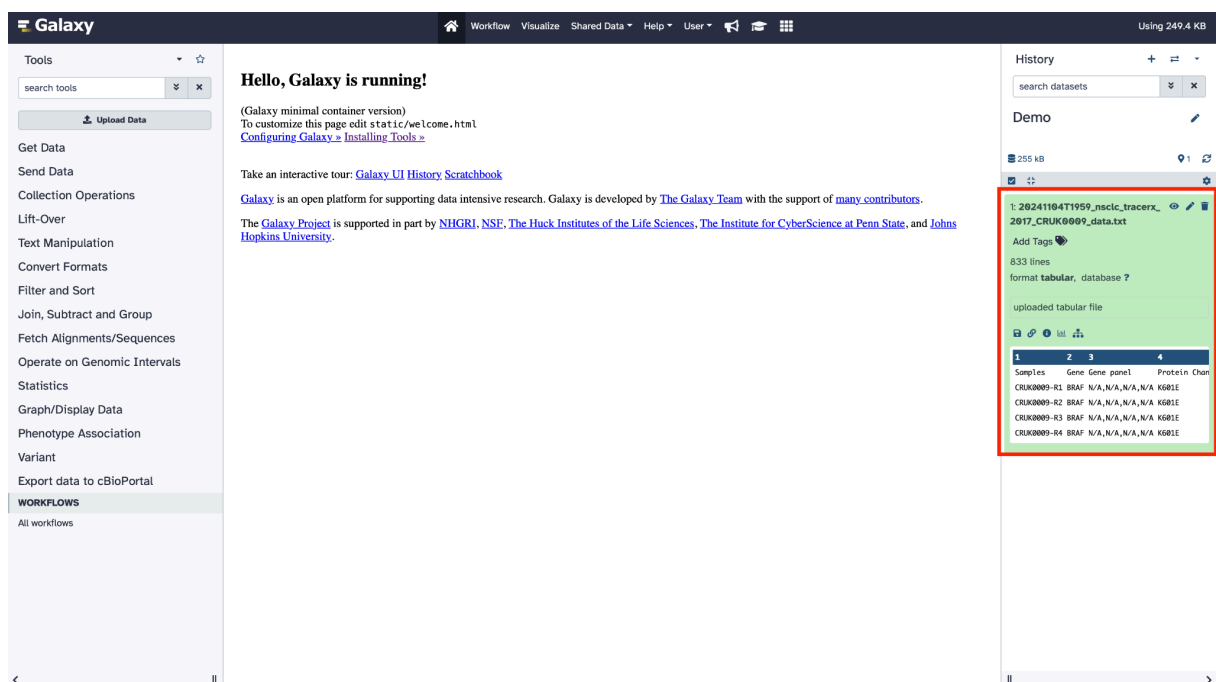


Figure 6. Data exported from cBioPortal with name data_studyID_casID

The mutation data are imported as a Galaxy data type that is similar to a table, meaning that any Galaxy tools handling tables are compatible with the imported data.

The next step is preparing data for PyClone-VI as it required a very specific format:

1. **mutation_id** - Unique identifier for the mutation. This is free form, but should match across all samples.
2. **sample_id** - Unique identifier for the sample.
3. **ref_counts** - Number of reads matching the reference allele.
4. **alt_counts** - Number of reads matching the alternate allele.
5. **major_cn** - Major copy number of segment overlapping mutation.
6. **minor_cn** - Minor copy number of segment overlapping mutation.
7. **normal_cn** - Total copy number of segments in healthy tissue. For autosomes this will be two and male sex chromosomes one.

Optional columns can be provided. For more information, see PyClone-VI GitHub repo or publication.

Once the data is in the correct format, run PyClone-VI, which can be found under the Variant tab.

Certain modifications have been made to the PyClone VI source code to enable its optimal use in Galaxy. Specifically, the management of parameters in the form of arguments given to the script has been implemented into a main function replacing the Click Python package, which is designed for creating command line interfaces in a composable way with as little code as necessary. This modification enables the user to define all the parameters in the form page of the tool in Galaxy (Fig.4), necessary for the correct operation of the tool.

The results of PyClone can be plotted with the Cluster Prevalence Over Samples tool under the Variant tab.

The screenshot displays the Galaxy web interface for the PyClone-VI tool. The left sidebar shows the 'Variant' tab selected, with a red box highlighting the 'Cluster Prevalence Over Samples' tool. The main panel shows the PyClone-VI tool configuration page with the following parameters:

- num_clusters**: 10 (Number of clusters to use in variational approximation distribution. Note that not all clusters may not be assigned data points, so the final number of clusters could be lower. Default is 10. (--num_clusters))
- Select number of annealing steps**: 1 (Number of simulated annealing steps to use. Default is 1. (--num_annealing_steps))
- Select number of points used to approximate CCF values**: 100 (Default is 100. (--num_grid_points))
- Select number of random restarts of variational inference**: 1 (Default is 1. (--num_restarts))
- Select exponent of entries in the annealing ladder**: 1.0 (Default is 1.0. (--annealing_power))
- Select maximum relative ELBO difference between iterations to decide on convergence**: 1e-05 (Default is 10^-6. (--convergence_threshold))
- Select number of ELBO optimization iterations**: 1

The right sidebar shows the history with two items: '2: Cluster Prevalence Over Samples_on_data 1.png' and '1: pyclone_vi_on_data_34_outfile.tsv'.

Figure 7. New Galaxy tools

4.1.4 Exporting data from Galaxy to cBioPortal

Two types of data can be exported back to cBioPortal:

- ▶ Images
- ▶ Timeline

The connection between the Galaxy instance and cBioPortal is defined by the administrator during the server setup. This configuration ensures that the Galaxy instance can communicate with the specified cBioPortal server. Detailed information about the connection setup is provided in the methods [section](#).

Steps to export images to cBioPortal:

1. **Tool Location:** The "Export Resource Image to cBioPortal" tool can be found under the Resource tab in Galaxy.

2. **Input Required:**

- ▶ **Galaxy history item (image):** Select the image generated during the analysis workflow.
- ▶ **Study ID:** Must match an existing study ID in cBioPortal.
- ▶ **Case ID:** The ID of the patient.
- ▶ **Image Name:** The name that the image will have on the intermediary server.
- ▶ **Overwrite:** If a file already exists, an error will be thrown unless you select this option.

3. **Output:**

- ▶ The image is hosted on the intermediary server and linked within cBioPortal as resource data.
- ▶ Used metadata and data files used to upload data are generated.

Steps to export timeline data to cBioPortal:

1. **Tool Location:** The "Export Timeline Data to cBioPortal" tool can be found under the Timeline tab in Galaxy.

2. **Input Required:**

- ▶ **PyClone-VI Output:** The TSV file generated by PyClone-VI.
- ▶ **Study ID:** Must match an existing study ID in cBioPortal.
- ▶ **Case ID:** The ID of the patient.
- ▶ **Each Time Point:**

- ▶ *Time-point ID*: Should be the same as the `sample_id` column from the PyClone-VI output.
- ▶ *Start Date in Days*: Numerical value indicating the start date.
- ▶ *End Date in Days*: (Optional) Numerical value indicating the end date.

4. Output:

- ▶ Timeline data is incorporated within cBioPortal, providing contextual analysis alongside other clinical and mutation data.
- ▶ Necessary metadata and data files are generated and uploaded:

These steps allow users to seamlessly export analytical results from Galaxy back into cBioPortal, thereby integrating workflow outputs with patient data in a comprehensive manner.

4.2 Live demonstration

A live demonstration has been set up to showcase the project. The demonstration is hosted on NREC¹⁶, a cloud infrastructure based on OpenStack¹⁷, which is a collaboration project between the University of Bergen and the University of Oslo. Additional sponsorship is provided by NeIC¹⁸ (Nordic e-Infrastructure Collaboration).

This live demonstration includes a modified instance of cBioPortal and a Galaxy server with the necessary tools to reproduce the workflows described in the user manual.

Several studies have been pre-loaded to demonstrate the project's capabilities. Please be mindful of resource usage as this demonstration environment is shared.

4.2.1 Access and registration

To access the demonstration:

- ▶ **cBioPortal Server**: <https://cbioportal.uio-elixir-devops.uiocloud.no>
- ▶ **Galaxy Server**: <https://galaxy.uio-elixir-devops.uiocloud.no>. Notes: Users must register to run the tools.

4.2.2 Security Considerations

All data related to this demonstration is securely hosted in Oslo within a trusted infrastructure. Personal data is not shared or stored permanently and is regularly wiped to safeguard privacy. Communication between servers and users is secured using HTTPS encryption. This encryption is managed by Caddy¹⁹, a reverse proxy, which helps ensure that all data transmitted is protected against interception.

¹⁶ <https://nrec.no>

¹⁷ <https://www.openstack.org>

¹⁸ <https://neic.no>

¹⁹ <https://caddyserver.com>

4.3 Methods

4.3.1 Current limitations

Currently, the only programmatic method for users to extract data from cBioPortal is via its API. This limitation means that exporting data directly from the portal interface is not possible. This poses a challenge for users who need to transfer data for further analysis using other platforms such as Galaxy. Moreover, it is not possible for users to upload data to cBioPortal programmatically via the API, further complicating the data flow between these platforms.

4.3.2 Proposed Solution

To address these limitations, we have developed an export feature specifically for mutation tables within the cBioPortal front-end code. This new functionality introduces an export button on cBioPortal that initiates an asynchronous request to send data to a designated API endpoint. To bring data back to cBioPortal, we have utilised a script developed by the cBioPortal team that uploads data using SQL procedures²⁰.

4.3.3 Data Extraction from cBioPortal

Overview

A fork of the cBioPortal front-end repository was created to develop this proof-of-concept. We reused as many existing functionalities as possible. To integrate our modifications into the official cBioPortal repository, we plan to open a Request for Comments (RFC) in the future²¹.

API Usage

The Galaxy button sends a POST request containing five pieces of information: Galaxy token, Galaxy history name, Study ID, Case ID, and raw table data to an API endpoint. This API endpoint is configurable in the cBioPortal configuration files using the **export_server_url** option.

The default API endpoint is **<http://localhost:3001/export-to-galaxy>**, built specifically for this project. However, any endpoint that can handle POST requests and return a 200 status code will suffice. It should be noted that this API endpoint is not the Galaxy server directly, but an intermediary server that processes the user's Galaxy information to export the data to Galaxy.

Data Export Functionality

The "Export Data to Galaxy" button was added as an extension of the CopyDownloadControls already present in the cBioPortal code²². In practice, the button calls a function that opens a modal pop-up to prompt the user for their Galaxy API key and history name. Once this information is provided, no further user input is needed. This function will use the same methods to extract mutation data from the cBioPortal database as the copy and download button. It then creates a POST request with a JSON payload containing the Galaxy API key, Galaxy history name, Study ID, Case ID, and the content of the mutation table in tab-delimited string format to the previously defined API endpoint.

²⁰ <https://docs.cbioportal.org/using-the-metaimport-script>

²¹ <https://docs.cbioportal.org/rfc-list>

²² <https://github.com/cBioPortal/cbioportal-frontend/tree/master/src/shared/components/copyDownloadControls>

Below is an example of a POST request in json format made by this functionality:

```
{
  "method": "POST",
  "url": "http://localhost:3001/export-to-galaxy/",
  "headers": {
    "accept": "application/json",
    "Content-Type": "application/json"
  },
  "body": {
    "galaxyToken": "your_galaxy_token",
    "galaxyHistoryName": "your_galaxy_history_name",
    "studyId": "your_study_id",
    "caseId": "your_case_id",
    "data": "your_data_string"
  }
}
```

4.3.4 Intermediary Server

Overview

The intermediary server acts as a bridge between cBioPortal and Galaxy, managing the data transfer and processing required for seamless integration. The server is built using FastAPI²³, a high-performance web framework for building APIs with Python 3.8+. Python was chosen for convenience, as the Galaxy project has developed a Python library, BioBlend²⁴, to simplify interactions with the Galaxy API. However, other API packages or languages could be used as well.

This server was developed specifically to bridge cBioPortal and Galaxy for multiple time points analysis, but can be expanded for other functionalities by adding API endpoints or implementing more robust data formatting or filtering.

It is recommended that the intermediary server is hosted on the same physical server as the cBioPortal server to address certain latency and security considerations, which will be discussed later.

Functionalities

API Endpoints: The application defines several API endpoints using FastAPI, which are organised into routers:

- ▶ **/upload-image/:** Handles operations related to images in Galaxy.
- ▶ **/export-timeline-to-cbioportal/:** Manages timeline data transfer from Galaxy to cBioPortal.
- ▶ **/export-resource-to-cbioportal/:** Manages resource data transfer from Galaxy to cBioPortal.
- ▶ **/export-to-galaxy/:** Manages data transfer from cBioPortal to Galaxy.

Environment Variables: The application retrieves and uses environment variables for configuration. These include:

- ▶ **GALAXY_URL:** URL for Galaxy.

²³ <https://fastapi.tiangolo.com/>

²⁴ <https://github.com/galaxyproject/bioblend>

- **CBIOPORTAL_URL**: URL for cBioPortal.
- **STUDY_DIRECTORY**: Directory where studies are stored for the cBioPortal server.
- **CBIOPORTAL_CACHE_API_KEY**: cBioPortal API cache key, defined in the cBioPortal configuration file.

Logging: Logging is configured with uvicorn²⁵, the ASGI server used to run FastAPI applications.

Export Data from cBioPortal to Galaxy

To export data to Galaxy, we use BioBlend, an open-source library created and maintained by the Galaxy project. It allows multiple ways to connect and interact with Galaxy via its API. For this proof-of-concept, we used an API key to identify and authorise users. Email and password authentication can also be used. We refer the reader to the BioBlend documentation²⁶ for more information.

This endpoint expects table mutation data to be a string with columns, separated by tabulation and rows by newline, as it does not format this data before sending it to Galaxy.

Export Data from Galaxy to cBioPortal

To import data from Galaxy to cBioPortal, we use the **metalimport.py** script from the cBioPortal-core repository²⁷. The main advantage of using this script is that it is maintained and synchronised with the current cBioPortal version. It also includes a data validator to prevent database corruption.

One issue with using any script from cBioPortal-core, including the **metalimport.py** script, is that the server needs read access to the cBioPortal configuration file **application.properties**²⁸, which can lead to security issues as this file contains sensitive data.

Currently, we have two API endpoints for uploading two types of data from Galaxy to cBioPortal²⁹:

- **/export-timeline-to-cbioportal/**: For timeline data³⁰.
- **/export-resource-to-cbioportal/**: For resource data³¹. Refer to the cBioPortal documentation for more information on how these data types are displayed.

The **/export-resource-to-cbioportal/** endpoint expects the POST request format:

```
{
  "method": "POST",
  "url": "http://localhost:3001/export-ressource-to-cbioportal/",
  "headers": {
    "accept": "application/json",
    "Content-Type": "application/json"
  },
  "body": {
    "dataDefinitionContent": "your_data_definition_content",
    "metaDefinitionContent": "your_meta_definition_content",
    "dataPatientContent": "your_data_patient_content",
    "metaPatientContent": "your_meta_patient_content",
  }
}
```

²⁵ <https://www.uvicorn.org/>

²⁶ <https://bioblend.readthedocs.io/en/latest/>

²⁷ <https://github.com/cBioPortal/cbioportal-core>

²⁸ <https://github.com/cBioPortal/cbioportal-core?tab=readme-ov-file#configuring-application-properties>

²⁹ https://github.com/elixir-oslo/cbioportal-galaxy-connector/blob/main/app/routers/galaxy_to_cbioportal_handler.py

³⁰ <https://docs.cbioportal.org/file-formats/#timeline-data>

³¹ <https://docs.cbioportal.org/file-formats/#resource-data>

```

    "studyId": "your_study_id"
  }
}

```

The **/export-timeline-to-cbioportal/** endpoint expects the POST request format:

```

{
  "method": "POST",
  "url": "http://localhost:3001/export-timeline-to-cbioportal/",
  "headers": {
    "accept": "application/json",
    "Content-Type": "application/json"
  },
  "body": {
    "dataContent": "your_data_content",
    "metaContent": "your_meta_content",
    "studyId": "your_study_id",
    "caseId": "your_case_id",
    "suffix": "your_suffix"
  }
}

```

The two main arguments of the scripts are the cBioPortal URL and the path to the data to be uploaded. The intermediary server can be hosted anywhere as long as it can connect to the desired cBioPortal instance.

For both those API endpoints, the Content variable needs to be already formatted for cBioPortal import, following the cBioPortal documentation.

There are two main ways to load data to cBioPortal: incremental upload and full study upload.

Incremental Upload³²: Preferred as it can upload specific data types to an already existing study without re-uploading the entire study. This means you do not need access to cBioPortal storage to add new data files. However, only certain cBioPortal data types support incremental upload. We use this method for timeline data.

Full Study Upload³³: Used for resource data (e.g., externally hosted data) that does not support incremental upload. This method validates and replaces the entire study.

For resource data, especially images, cBioPortal does not store the resources themselves, but links to hosted resources. Since the images are produced by Galaxy and are not meant for permanent storage, we added an image hosting function on the intermediary server. Images can be uploaded to the intermediary server and accessed at a specific address: **http://<intermediary_server_url>/images/<image_name>**. This address is returned as an API response. This ensures that images remain accessible even if the Galaxy server is down or the original image is removed. The **upload-image/** API endpoint is used and expects the following POST request format:

```

{
  "method": "POST",
  "url": "http://localhost:3001/upload-image/",
  "headers": {
    "accept": "application/json",
    "Content-Type": "multipart/form-data"
  },
  "formData": {
    "file": "path_to_your_image_file",
    "overwrite": "false"
  }
}

```

³² <https://docs.cbioportal.org/incremental-data-loading/>

³³ <https://docs.cbioportal.org/data-loading/#loading-data>

```
}
}
```

Since the intermediary needs direct access to the cBioPortal study directory, we added external validation for the uploaded data. If data concerning the patient already exists, the new data will replace the old data. For timeline data, a suffix variable allows multiple data points for the same patient to coexist.

Finally, to display modifications to the cBioPortal database on the portal, the server needs to be restarted. However, if the cBioPortal instance was deployed to cache information on the backend to improve performance, a simple clearing of this cache is sufficient. We are leveraging this solution by sending a cache eviction DELETE request to the cBioPortal endpoint `/api/cache`³⁴. This process could be improved in the future, as it is possible to only evict cache for a single study, but this was not implemented in this version. Please note that an API key defined in the cBioPortal configuration file is necessary to evict this cache. It is configured by the environment variable `CBIOPORTAL_CACHE_API_KEY`. For this proof-of-concept, we use Redis³⁵ as the backend caching option. Refer to the cBioPortal documentation for more information on setting up and cache eviction³⁶.

4.3.5 Galaxy integration

Overview

Galaxy is an open, web-based platform for accessible, reproducible, and transparent computational research, particularly in the field of biomedical research. It provides a user-friendly interface where researchers can access a vast array of bioinformatics tools and workflows. Galaxy enables users to perform complex computational analyses without the need for in-depth programming knowledge. Key features of Galaxy include:

- ▶ **Ease of Use:** Galaxy is designed to be user-friendly, with a graphical interface that allows users to design and execute complex analysis workflows.
- ▶ **Reproducibility:** Galaxy maintains a detailed history of every analysis, capturing all parameters and tools used, which ensures results can be reproduced.
- ▶ **Accessibility:** Galaxy is accessible through a web browser, eliminating the need for local installation and configuration of bioinformatics tools.
- ▶ **Scalability:** Galaxy can be run on individual laptops, in powerful server clusters, or in cloud-based environments, making it adaptable to a wide range of computational needs.

For more in-depth information on Galaxy, refer to the official Galaxy documentation³⁷.

Challenges in Automating Workflows

One of the primary challenges in automating workflows in Galaxy is that the data input from cBioPortal is not standardised. This non-standardisation can cause various complications:

- ▶ **Data Format Variability:** Data exported from cBioPortal may vary in formatting, structures, and types, making it difficult to create a one-size-fits-all workflow in Galaxy.
- ▶ **Metadata Inconsistencies:** Metadata associated with mutation tables and other data types may be inconsistent, which can affect the processing and analysis steps.

³⁴ <https://docs.cbioportal.org/deployment/customization/caching/>

³⁵ <https://redis.io/>

³⁶ <https://docs.cbioportal.org/deployment/customization/caching/#cache-eviction>

³⁷ <https://docs.galaxyproject.org/>

- **User-Defined Parameters:** Users may input different parameters or have different expectations for their analyses, requiring flexible and adaptable workflows.

These challenges necessitate additional preprocessing steps, custom scripts, and adaptable tools to handle the variability and ensure successful integration and meaningful analysis.

When you control the input format, Galaxy workflows can be used. Moreover, these workflows can be launched using BioBlend³⁸. It means that if the mutation table data is formatted as expected by the workflow, the data can be sent to Galaxy, analysed, and sent back to cBioPortal with no user input required after they provide their Galaxy API key and history name.

Created Tools

To address these challenges and to facilitate the integration of cBioPortal data with Galaxy, we developed several tools. Each tool is designed to perform specific tasks within the overall workflow, enhancing the functionality and adaptability of the integration.

Wrapper for PyClone-VI

The Pyclone-VI tool has been introduced into the Galaxy universe in the form of a wrapper. The process of depositing a tool on the Galaxy Toolshed under the IUC (Intergalactic Utilities Commission) involved several steps to ensure that the tool meets quality standards and integrates seamlessly into Galaxy. Consequently, a Galaxy tool wrapper specifies the tool's inputs, outputs, and parameters in XML. The tool has been tested within a local Galaxy instance to ensure functionality and compliance with Galaxy standards. In order to integrate the IUC toolshed, we have followed the development standards recommended by the Galaxy development team and the best practices³⁹. These include using standardised XML formatting, handling dependencies properly (e.g., Conda⁴⁰ or Bioconda⁴¹), and ensuring compatibility with Galaxy. Following this structured process ensures your tool to be robust, well-documented, and aligned with the Galaxy ecosystem.

We also used Planemo, which is a tool to assist with Galaxy tool development and validation. It helps to lint your tool for errors, write tests and run tests in an isolated environment.

The tool is still in the reviewing process and should be available in December 2024 on Galaxy Europe⁴². Once approved, the tool will be made available in the Toolshed for installation by Galaxy administrators worldwide. For the purposes of the demo, the wrapper was integrated into the docker containing the Galaxy instance.

Plotting for PyClone-VI Output

- **Name:** Cluster Prevalence Over Samples
- **Description:** This Python-based script plots the prevalence of clusters over samples from the TSV output of PyClone-VI.
- **Input:** PyClone-VI tab-separated output data.
- **Process:** It processes the output data and creates a line plot where each line represents a cluster.
- **Output:** High-quality graphical representations of the clonal populations.

³⁸ https://bioblend.readthedocs.io/en/latest/api_docs/galaxy/docs.html#invoke-a-workflow

³⁹ https://galaxy-iuc-standards.readthedocs.io/en/latest/best_practices/integration_checklist.html

⁴⁰ <https://docs.conda.io/projects/conda/en/latest/index.html>

⁴¹ <http://bioconda.github.io/>

⁴² <https://github.com/galaxyproject/tools-iuc/pull/6144>

Export Resource Image to cBioPortal

- ▶ **Name:** Link image to cBioPortal resource
- ▶ **Description:** This Python-based tool exports any images generated during Galaxy workflows back to cBioPortal as resource data.
- ▶ **Input:** Path to the generated images in Galaxy.
- ▶ **Options:**
 - ▶ *Study ID:* (must match an existing study ID)
 - ▶ *Case ID:* ID of the patient.
 - ▶ *Image Name:* Name that the image will have on the intermediary server.
 - ▶ *Overwrite:* If a file already exists, an error will be thrown unless this option is selected.
- ▶ **Process:** It first uploads the image to the intermediary server using the **/upload-image/** API endpoint. With the response from the first request, the script will format the information into four necessary files for resource data for a cBioPortal Import. The content of these files is then sent to the **/export-resource-to-cbioportal/** endpoint.
- ▶ **Output:** The following files used to export data to cBioPortal:
 - ▶ Metadata definition file
 - ▶ Data definition file
 - ▶ Metadata resource patient file
 - ▶ Data resource patient file

Export Timeline to cBioPortal

- ▶ **Name:** Load PyClone-VI output to cBioPortal Timeline
- ▶ **Description:** This tool exports clonal population data generated by PyClone-VI back to cBioPortal as timeline data.
- ▶ **Input:** PyClone-VI tab-separated output.
- ▶ **Options:**
 - ▶ *Study ID:* (must match an existing study ID)
 - ▶ *Case ID:* ID of the patient.
 - ▶ For each timepoint:
 - *Time-point ID:* Same as the sample_id column from PyClone-VI output.
 - *Start Date in Days:* Numerical value indicating the start date.
 - *End Date in Days:* (Optional) Numerical value indicating the end date.
 - *Process:* It formats the PyClone-VI data into cBioPortal-compatible files and uploads them to the intermediary server, which then integrates them into cBioPortal.
- ▶ **Output:** The following files used to export data to cBioPortal:

- ▶ Metadata timeline file
- ▶ Data timeline file

4.3.6 Code Availability and Deployment Instructions

GitHub Repositories

All components of this project, including the modified cBioPortal front-end and portal, the intermediary server, the individual Galaxy tools, and a Galaxy instance with the necessary tools, are hosted on GitHub. Below are the links to each repository and a brief description:

Modified cBioPortal Front-End Repository:

- ▶ **Description:** Contains the modified cBioPortal front-end and any associated changes made to support the new export functionality.
- ▶ **Link:** <https://github.com/elixir-oslo/cbioportal-frontend>

Modified cBioPortal Project Repository:

- ▶ **Description:** Contains the modified cBioPortal repository to build the full project using the modified cBioPortal front-end.
- ▶ **Link:** <https://github.com/elixir-oslo/cbioportal> (branch tmr)

Intermediary Server Repository:

- ▶ **Description:** Hosts the intermediary server code built using FastAPI, which manages data transfer between cBioPortal and Galaxy.
- ▶ **Link:** <https://github.com/elixir-oslo/cbioportal-galaxy-connector>

Galaxy with Integrated Tools Repository:

- ▶ **Description:** Contains the Docker files and configuration to create a Galaxy instance container with all necessary bioinformatics tools pre-installed, along with Docker setup for easy deployment.
- ▶ **Link:** <https://github.com/elixir-oslo/docker-galaxy-pyclone>

Full Project Deployment Repository:

- ▶ **Description:** A branch of the cbioportal-docker-compose repository, modified to include all components (cBioPortal, intermediary server, and Galaxy) for running the full project on a server.
- ▶ **Link:** <https://github.com/elixir-oslo/cbioportal-docker-compose> (branch tmr)

Python Script and Galaxy Tool Repositories:

- ▶ **Wrapper for PyClone-VI:**
 - ▶ **Description:** A Galaxy tool wrapper designed to run the PyClone-VI software for clonal population inference.
 - ▶ **Link:** <https://github.com/elixir-oslo/galaxy-tool-pyclone-vi>

► **Plotting for PyClone-VI Output:**

- **Description:** A Galaxy tool that takes PyClone-VI output and generates visual plots.
- **Link:** <https://github.com/elixir-oslo/galaxy-tool-plot-cluster-prevalence>

► **Export Resource Image to cBioPortal:**

- **Description:** A Python script and Galaxy tool that exports images generated in Galaxy back to cBioPortal.
- **Link:** <https://github.com/elixir-oslo/galaxy-tool-export-cbioportal-image>

► **Export Timeline Data to cBioPortal:**

- **Description:** A Galaxy tool that exports PyClone-VI data in timeline format back to cBioPortal.
- **Link:** <https://github.com/elixir-oslo/galaxy-tool-export-cbioportal-timeline>

Docker and GitHub Actions

Each component of this project (Modified cBioPortal, Intermediary Server, and Project Galaxy Server) is containerized using Docker⁴³, with GitHub Actions⁴⁴ set up to build and publish the Docker images repository⁴⁵. This ensures a streamlined and reproducible deployment process.

► **Dockerfiles:**

- Each repository contains a **Dockerfile** that specifies how to build the Docker image for that component.
- You can find the **Dockerfile** in the root directory of each repository.

► **GitHub Actions:**

- GitHub Actions workflows are used to automate the building and publishing of Docker images and are automatically run for each new version.
- These workflows are defined in the **.github/workflows** directory of each repository.
- Images are pushed to the GitHub Container Registry (GHCR).

Running the Full Project

To run the full project on a server, you can use the modified **cbioportal-docker-compose** repository. This repository includes all the necessary elements to deploy cBioPortal, the intermediary server, and Galaxy in a single environment using Docker Compose. It must be noted that this Docker Compose setup should not be run in a production setting as is, as communications are not secured via HTTPS, and further steps must be taken to ensure that cBioPortal and Galaxy are running using their recommended configurations.

⁴³ <https://docs.docker.com/>

⁴⁴ <https://docs.github.com/en/actions>

⁴⁵ <https://docs.github.com/en/packages/working-with-a-github-packages-registry/working-with-the-container-registry>

Instructions:

1. Clone the Repository:

```
git clone https://github.com/elixir-oslo/cbioportal-docker-compose.git
cd cbioportal-docker-compose
```

2. Configuration:

- ▶ Images used are defined in the **.env** file.
- ▶ cBioPortal must be configured using the **application.properties** file that should be present in the **config** directory by default. An example file is included in the repo. You must configure your intermediary server API endpoint for exporting in this file by adding the following option:
`export_server_url=http://<cbioportal-galaxy-connector-url>/export-to-galaxy`. Please note that cBioPortal will contact the intermediary server via your web browser, so the `<cbioportal-galaxy-connector-url>` cannot be a Docker URL based on the container name.
- ▶ As described in the methods, the intermediary server needs access to the **application.properties** file and **study** directory, which both need to be mounted as Docker volumes. Set your Galaxy and cBioPortal instance URLs. Additionally, the following environment variables need to be set:
 - ▶ **GALAXY_URL**: Galaxy instance URL where data will be exported.
 - ▶ **CBIOPORTAL_URL**: cBioPortal instance URL where data will be exported.
 - ▶ **CBIOPORTAL_CACHE_API_KEY**: cBioPortal cache API key defined in the **application.properties** file (see Redis configuration for more information).
- ▶ The project-ready Galaxy instance is based on version 23.2.1 but contains all the tools created for this project. It must be configured with the following environment variables:
 - ▶ **EXPORT_TIMELINE_ENDPOINT**: Intermediary API endpoint for uploading timeline data to cBioPortal.
 - ▶ **UPLOAD_IMAGE_ENDPOINT**: Intermediary API endpoint for uploading an image to this server.
 - ▶ **CBIOPORTAL_LOAD_RESOURCE_ENDPOINT**: Intermediary API endpoint for uploading resource data to cBioPortal.
 - ▶ **IMAGE_BASE_URL**: URL where images should be accessible.
- ▶ The Redis server must be configured with a **redis.conf** file. An example file is provided in the repo. cBioPortal must also be configured to use cache by modifying the **application.properties** files as followed:

```
persistence.cache_type=redis
cache.endpoint.api-key=<random-api-key>
redis.name=cbioportal_galaxy
redis.leader_address=redis://<redis-leader-url>:<redis-port>
redis.follower_address=redis://<redis-follower-url>:<redis-port> # Can be the
same as redis.leader_address if you only have one redis cache
redis.database=0
redis.password=<redis-password>
redis.ttl_mins=10000
redis.clear_on_startup=true
```

3. Start the Services:

```
docker-compose up -d
```

4. Access the Services:

- ▶ cBioPortal: `http://localhost:8080` (default)
- ▶ Intermediary Server: `http://localhost:3001` (default)
- ▶ Galaxy: `http://localhost:8081` (default, cannot be changed in the Docker Compose setup)

Individual Component Documentation

Each repository contains documentation to run and test the components individually. Refer to the `README.md` files in each repository for detailed instructions:

- ▶ **Modified cBioPortal:** Documentation on setting up and running the modified cBioPortal instance.
- ▶ **Intermediary Server:** Documentation on configuring and deploying the intermediary server using FastAPI.
- ▶ **Galaxy with Integrated Tools:** Instructions on running the Galaxy instance with the pre-installed tools, including sample workflows and usage examples.

4.3.7 Security Considerations

- ▶ **Security Measures to Protect Data During Transfer:**
 - ▶ Since every communication between the intermediary server and the cBioPortal and Galaxy instances is in plain text by default, it is imperative to secure these endpoints using the HTTPS protocol. This involves setting up SSL/TLS certificates for the intermediary server, cBioPortal, and Galaxy instances.
 - ▶ To further secure the process, data encryption should be implemented at every point of communication. This includes encrypting data before sending it over the network and decrypting it upon receipt.
- ▶ **User Authentication and Data Access Controls:**
 - ▶ Both Galaxy and cBioPortal can be configured to use robust user authentication mechanisms, such as integrating with the authentication service Keycloak^{46 47}.
 - ▶ Keycloak identity provider can manage user authentication and authorization, providing Single Sign-On (SSO) capabilities and fine-grained access control.
 - ▶ It is crucial to implement Role-Based Access Control (RBAC) to ensure that only authorised users have access to sensitive data and functionalities within cBioPortal and Galaxy.
 - ▶ Ensure that API keys and other credentials are stored securely, and that access is restricted to authorised users only.

By organising code repositories and deployment instructions clearly and systematically, users can easily access, deploy, and contribute to the project. This setup ensures that the integration of cBioPortal and Galaxy is both reproducible and scalable, facilitating broader usage and collaboration.

⁴⁶ <https://docs.cbioportal.org/deployment/authorization-and-authentication/authenticating-and-authorizing-users-via-keycloak/>

⁴⁷ <https://galaxyproject.org/blog/2020-02-gv15-beta/>

4.4 Discussion

4.4.1 Key benefits

- ▶ **Simplified Data Workflow:** Users can easily export, analyse, and re-import data without requiring extensive manual procedures. This reduces time and potential errors associated with manual data handling.
- ▶ **Enhanced Analytical Capabilities:** Leveraging Galaxy's powerful tools enhances the depth and quality of analysis performed on mutation data from cBioPortal. This can lead to more insightful results and better patient outcomes.
- ▶ **Modular and Flexible Design:** The intermediary server allows for custom development, facilitating tailored data handling and integration solutions. This modular approach allows for future enhancements and adaptability.

4.4.2 Future considerations

- ▶ **Automation and Scheduling:** Integration of automated scheduling for regular data transfers and analyses. This will reduce the need for manual intervention and ensure timely data processing.
- ▶ **User Authentication and Permissions:** Implementing robust authentication and permission management to ensure secure data transfer and processing. This includes potentially integrating with existing user management systems.
- ▶ **Extended Analysis Options:** Expanding the range of analytical tools and workflows available in Galaxy for use with cBioPortal data. This can include incorporating new methods for different types of genomic data analysis. Making the code open source should favorise future improvements.
- ▶ **Usability improvements:**
 - ▶ Currently, there is no way for users to know where the data will be sent, either from cBioPortal to Galaxy or Galaxy to cBioPortal. To improve user experience, displaying the destination Galaxy or cBioPortal server within their respective interfaces would be desirable.
 - ▶ Galaxy instance destination for cBioPortal data is set at server start and cannot be changed otherwise. A future consideration would be to allow users to choose the Galaxy instance they want to send the data to, provided they have an account and an API key for this instance.

5 Conclusion

The demonstrations and descriptions provided herein clearly show that it is both possible and feasible to realise federated analysis across data repositories of sensitive human data, such as cancer data. The requirement is that data representations are standardised, with well adopted standards for each data type. Portability of software is a prerequisite, and this also requires care for solutions to work in shielded environments such as the Trusted Research Environment where cancer data often reside. This is important, as installation and maintenance must be possible even with limited internet access.