SOFTWARE TOOL ARTICLE

Sherlock: an open-source data platform to store, analyze and integrate Big Data for biology [version 1; peer review: 2 approved with reservations]

Balazs Bohar^{1,2}, David Fazekas^{1,2}, Matthew Madgwick^{1,3}, Luca Csabai^{1,2}, Marton Olbei^{1,3}, Tamás Korcsmáros^{id}¹⁻³, Mate Szalay-Beko¹

¹Earlham Institute, Norwich Research Park, Norwich, UK

²Department of Genetics, Eotvos Lorand University, Budapest, Hungary

³Gut Microbes and Health Programme, Quadram Institute Bioscience, Norwich Research Park, Norwich, UK

v1 First published: 21 May 2021, 10:409
<https://doi.org/10.12688/f1000research.52791.1>
 Second version: 10 Aug 2022, 10:409
<https://doi.org/10.12688/f1000research.52791.2>
 Latest published: 12 Jan 2023, 10:409
<https://doi.org/10.12688/f1000research.52791.3>

Abstract

In the era of Big Data, data collection underpins biological research more so than ever before. In many cases this can be as time-consuming as the analysis itself, requiring downloading multiple different public databases, with different data structures, and in general, spending days before answering any biological questions. To solve this problem, we introduce an open-source, cloud-based big data platform, called Sherlock (<https://earlham-sherlock.github.io/>). Sherlock provides a gap-filling way for biologists to store, convert, query, share and generate biology data, while ultimately streamlining bioinformatics data management. The Sherlock platform provides a simple interface to leverage big data technologies, such as Docker and PrestoDB. Sherlock is designed to analyse, process, query and extract the information from extremely complex and large data sets. Furthermore, Sherlock is capable of handling different structured data (interaction, localization, or genomic sequence) from several sources and converting them to a common optimized storage format, for example to the Optimized Row Columnar (ORC). This format facilitates Sherlock's ability to quickly and easily execute distributed analytical queries on extremely large data files as well as share datasets between teams. The Sherlock platform is freely available on Github, and contains specific loader scripts for structured data sources of genomics, interaction and expression databases. With these loader scripts, users are able to easily and quickly create and work with the specific file formats, such as JavaScript Object Notation (JSON) or ORC. For computational biology and large-scale bioinformatics projects, Sherlock provides an open-source platform empowering data management, data analytics, data integration and collaboration

Open Peer Review

Approval Status

	1	2
version 3		
(revision)		
12 Jan 2023		
version 2
(revision)	view	view
10 Aug 2022
version 1
21 May 2021	view	view

1. **John H. Morris** ^{id}, University of California, San Francisco, San Francisco, USA

2. **Nadezhda T. Doncheva** ^{id}, University of Copenhagen, Copenhagen, Denmark

Any reports and responses or comments on the article can be found at the end of the article.

through modern big data technologies.

Keywords

Software, Computational biology, Network biology, Systems biology, Data lake, big data, data management, data integration

This article is included in the **Artificial Intelligence and Machine Learning** gateway.

This article is included in the **Bioinformatics** gateway.

Corresponding author: Tamás Korcsmáros (t.korcsmaros@imperial.ac.uk)

Author roles: **Bohar B:** Conceptualization, Data Curation, Methodology, Software, Writing – Original Draft Preparation, Writing – Review & Editing; **Fazekas D:** Conceptualization, Data Curation, Methodology, Resources, Software, Writing – Review & Editing; **Madgwick M:** Methodology, Software, Writing – Review & Editing; **Csabai L:** Software, Writing – Review & Editing; **Olbei M:** Software, Writing – Original Draft Preparation, Writing – Review & Editing; **Korcsmáros T:** Conceptualization, Data Curation, Project Administration, Resources, Supervision, Writing – Review & Editing; **Szalay-Beko M:** Conceptualization, Data Curation, Methodology, Resources, Software, Supervision, Writing – Original Draft Preparation, Writing – Review & Editing

Competing interests: No competing interests were disclosed.

Grant information: MM is supported by a Biotechnological and Biosciences Research Council (BBSRC) funded Norwich Research Park Biosciences Doctoral Training Partnership (grant number BB/S50743X/1), as an NPIF Award. MO was supported by the BBSRC Norwich Research Park Biosciences Doctoral Training Partnership (grant BB/M011216/1). The work of T.K. and M.Sz.B was supported by the Earlham Institute (Norwich, UK) in partnership with the Quadram Institute (Norwich, UK) and strategically supported by the UKRI BBSRC UK grants (BB/J004529/1, BB/P016774/1, and BB/CSP17270/1). T.K. was also supported by a BBSRC ISP grant for Gut Microbes and Health BB/R012490/1 and its constituent projects, BBS/E/F/000PR10353 and BBS/E/F/000PR10355.

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

Copyright: © 2021 Bohar B *et al.* This is an open access article distributed under the terms of the **Creative Commons Attribution License**, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

How to cite this article: Bohar B, Fazekas D, Madgwick M *et al.* **Sherlock: an open-source data platform to store, analyze and integrate Big Data for biology [version 1; peer review: 2 approved with reservations]** F1000Research 2021, 10:409 <https://doi.org/10.12688/f1000research.52791.1>

First published: 21 May 2021, 10:409 <https://doi.org/10.12688/f1000research.52791.1>

Introduction

Most bioinformatics projects start with gathering a lot of data. Often bioinformaticians have to work on bespoke datasets, for example, gene expression or mutation data, but in almost all cases this requires some sort of external reference data. The vast majority of projects may call for genome annotations, gene ontologies, tissue-specific expression datasets, drug-related databases, and many other, particular reference datasets. One of the reasons why we use the term 'bioinformatics' in the first place is because we cannot correlate and process all these datasets manually, we need the help and the power of the computers and databases (Greene et al. 2014). Thanks to the current technical advancement, many solutions exist worldwide to take advantage of the available computational possibilities (Marx 2013).

Cloud storage solutions such as Amazon's S3 (<https://aws.amazon.com/s3/>) or Google Cloud Storage (<https://cloud.google.com/storage>) offer scalability and flexibility to the matching compute solutions. More importantly, they allow large and potentially shared datasets to be stored on the same infrastructure where large-scale analyses are run. Some companies have utilized cloud resources to offer storage, access to shared datasets, and transparent sharing of data. Cloud storage also provides enhanced reliability, as the data is backed up in several geographical locations (Kasson 2012). However, in most of the cases, these cloud storage companies only offer data storage solutions, but this does not include platforms to execute or work with data.

To address these issues, we repurposed concepts and open source tools from the top players of the software industry, like Facebook, Netflix and Amazon. With the aim to replace the tedious process of manual data collection before the first steps of any bioinformatics project, we developed a new platform, Sherlock. The Sherlock platform has two main parts: 1) a query engine, which is responsible for executing the given Structured Query Language (SQL) queries, which is the most extensively used database language; 2) a Data Lake, required for data storage, which consists of multiple different databases or datasets, where the Query engine can execute queries against the data. The combination of these two parts streamlines efficient data collection, data integration, and data preprocessing for a given project (Khine & Wang 2018).

Implementation and Operation

Overview

Sherlock is an open source and freely accessible platform that combines several different software technologies together (<https://earlham-sherlock.github.io/>). One of the core software that Sherlock uses is Docker. Docker is an open and widely used technology to isolate Linux processes and provide them a reproducible, well defined runtime environment independent from the actual operating system (Matthias & Kane 2015). Each element in the Sherlock platform is running inside a separate Docker container. A Docker container is a standard unit of software that packages up all of the necessary code and all the dependencies for them, so the application runs quickly and reliably in an isolated environment on any computer. Container orchestration tools are used when a complex solution (like Sherlock) requires the interconnection of many docker containers distributed among multiple separate Linux machines. In Sherlock, we are using the standard Docker Swarm orchestration platform (Smith 2017), as it is easy to use and much easier to operate than other orchestration frameworks (like Kubernetes (<https://kubernetes.io>)). These industry standard technologies make the management (starting, stopping and monitoring) of Sherlock easy, while also enabling us to implement more advanced features, for example the on-demand scaling of the analytical cluster where Sherlock is running. This scalability is the biggest advantage of Docker Swarm. The entire cluster can be shut down or reduced to the minimum in the cloud when it is not used, while it can be scaled up to hundreds of nodes if necessary for executing very complex analytical queries. Other key benefits associated with Docker Swarm is the high level of availability offered for applications. This has a hierarchical structure, where several worker nodes and at least one manager node is responsible for handling the worker nodes' resources and ensuring that the cluster operates efficiently.

The first step for using Sherlock is to start a Docker Swarm with the deployment scripts on a cluster with several worker nodes. These scripts start different services, but each in a separate container. These deployment scripts are configurable (Figure 1). One can specify exactly how many resources a service can use at once (e.g. CPU, memory allocation). Inside a running Docker Swarm, there are two different main parts of Sherlock: the Hive metastore and the Presto query engine. Sherlock follows the industry best practices in its architecture to enable scalability. Usually, the main idea behind scalable batch processing architectures is the separation of data storage and analytics (Dean & Ghemawat 2008). This architecture allows us to scale the analytical power and the storage size independently from each other and even dynamically. This is the case with Sherlock when deployed in the cloud.

The Sherlock platform does not include data, instead provides a platform where one can work on and manipulate the data (Figure 1). Then by leveraging powerful database architectures (Presto query engine and Hive metastore) and providing them a channel to connect to the Data Lake (where we are storing our data) the user can run different analytical queries on top of the data files. The starting step for using Sherlock is to submit an SQL query to the Presto query engine, which can answer the biological question in interest and then it will connect to the Data Lake and fetch the necessary data files. Then the final step is to execute the submitted SQL query on top of the data files and then it takes back the results to the user (Figure 1).

Figure 1. Overview of the Sherlock platform and its relationship with the Data Lake. In this image you can see how the different tools and the deployment scripts are connected to each other inside the core of Sherlock. The blue box represents the Data Lake, where the data is stored. You can see that only the Hive Metastore, the Presto Query Engine and the possible worker nodes have connection to the Data Lake. The box with the dashed line shows the Minio S3 server, where for example, the Sherlock platform can be used to test various features available in Sherlock.

The minimal requirements for Sherlock depend on what the use case is and where Sherlock will be deployed. Sherlock can be used on a single laptop, single virtual machine (VM) or on a distributed cluster. Regarding using Sherlock on a single laptop we recommend using a relatively powerful laptop, which means that it has at least a dual-core processor (CPU) and at least 12 GB memory (RAM), as around half of these resources can be consumed by Sherlock. But the minimum requirement is at least 8 GB RAM and a dual-core processor. On the other hand, if the user wants to use it on a cluster with nodes, we will assume to have at least two machines with 32 GB memory (RAM) for each, and each of them having 8 processor (CPU) cores, but this can be scaled down to 16 GB RAM and 6 CPU cores.

Sherlock as a platform

Query engine

The first core part of Sherlock is the Query Engine (Figure 2A), which is responsible for running the given question through SQL commands on top of the data files and retrieving the 'answer' for the user. Query Engines, for example Hive, Impala, Spark SQL or Presto, are distributed, mostly stateless and scalable technologies. In our case, we are using the Presto Query Engine (<https://prestodb.io>), developed by Facebook. Presto provides a high-performance Query Engine wrapped in an easily deployable and maintainable package. Furthermore, the Presto Query Engine can handle many different types of data storage solutions.

With Presto, we can formalize analytical questions using SQL. SQL is composed of three main sub-languages: 1) data definition language (DDL), which allows the specification of database schemas; 2) a data manipulation language (DML), which supports operations on data (store, modify, delete); 3) and a data control language (DCL), which enables database administrators to configure security access to databases. On the one hand SQL is designed for working data, which held in a relational database management system (RDBMS), on the other hand it can be used for stream processing in a relational data stream management system (RDSMS). Moreover it is particularly really useful to handle with structured data (Silva et al. 2016). When combined with a Query Engine, SQL can be used to connect to the Data Lake, read and combine the data stored within it and execute different analytical queries, either sequentially or in parallel depending on the power of the given computer.

Hive metastore

The second core part of Sherlock is the Hive metastore (<https://docs.cloudera.com/runtime/7.0.1/hive-metastore/topics/hive-hms-introduction.html>), which is a service that stores metadata related to Apache Hive and other services, in a backend RDBMS, such as MySQL or PostgreSQL. With regards to Sherlock, Presto stores the metadata about the folders

Figure 2. The relationship between the two parts of Sherlock: the platform itself with the query engine and the Data Lake. A: The structure of Sherlock data platform with the core part, which is the Presto Query Engine. The user can execute different analytical SQL queries with the help of this query engine. Presto will run these queries on top of the data files, which are inside the Data Lake (B) **B:** The structure of our Data Lake. You can see the four different zones inside the Data Lake, what we are using right now. 1) The raw zone with the raw data from the different external databases. 2) The landing zone, with the Presto compatible text files in JSON Lines format. 3) The master zone, where the data is in a detailed and exact format, called ORC. 4) The project zone, where we store only the data which is needed only for specific projects.

in the Data Lake and the Hive metastore which contains only the meta information about the different folders, which is in the Data Lake. In Sherlock we provide simple scripts to save this metadata in the Data Lake when you want to make a backup or before you want to terminate your analytical cluster.

Deployment

For Sherlock, we developed a dockerized version of Presto that also incorporates the Hive metastore. This addition of both the query engine and metastore makes Sherlock cloud-agnostic, so it can be installed at any cloud provider, providing it is a Linux machine with Docker installed. The advantage of running Presto in a dockerized environment is that it is not necessary to install and configure the whole Presto Query Engine manually. Furthermore, it can even be fired up on a local machine, multiple machines or any cloud service as well. On the deployment guide section of the GitHub page of Sherlock (https://earlham-sherlock.github.io/docs/deployment_guide.html), we show how to make a set of Linux virtual machines with Docker installed, then start a distributed Presto cluster by using the Sherlock platform.

The Data Lake - data storage

A Data Lake is a simple network storage repository that can be accessed by external machines (<https://aws.amazon.com/big-data/datalakes-and-analytics/what-is-a-data-lake/>). All the data which is imported into, transformed in, or exported from Sherlock will be stored here as simple data files. The biggest advantages of using a Data Lake is that its operation will be the same on a local machine, on the cloud and the data can be stored in a well structured way on a reliable storage. Furthermore it can be scaled independently from the query engine. The technologies in Sherlock are compatible with both of the most common Data Lake solutions; Hadoop Distributed File System (HDFS), which is a distributed file system designed to run on commodity hardware, and with the Simple Storage Service (S3) storage formats. However, we decided to only describe S3 in all of our examples, as S3 is more widely-used, modern, and accessible. Although HDFS is an older standard solution, it has some very powerful features which can result in better performance, but all in all it is also much more difficult to set up, maintain and in most cases its extra features are not necessary for the given project. In contrast, S3 is a standard remote storage API (Application Programming Interface) format, introduced first by Amazon (<https://aws.amazon.com/s3/>). As of writing, you can purchase S3 storage as a service from all major cloud providers like Digital Ocean, Amazon AWS, Google Cloud or Microsoft Azure and each of these can be compatible with the Sherlock platform.

Having somewhere to store the data is only one half of having an operational Data Lake. One cannot stress enough how important it is to organize the data in a well defined 'folder' structure. Many Data Lake deployments become unusable after a few years due to poor maintenance, resulting in a large amount of data 'lost' in the thousands of folders, or inconsistent file formats and no ownership over the data. Our solution is to separate all of the data in the Data Lake into four different main folders, representing different stages of the data and different access patterns in the Data Lake. Inside each of these main folders we can create subfolders, and it is a good practice to incorporate the name of the dataset, the data owner name, the creation date (or other version info), and the file formats somehow into their paths.

We separated our Data Lake into four different main zones, which are built on top of each other (Figure 2B). The first zone is the raw zone. Into the raw zone, we archived all the database files in their original formats. For example: if we downloaded the human genome, then we put the fasta files here, under a separate subfolder. The name of the subfolder should contain the exact version (for example: hg38_p12) (Figure 3), and also we put a small readme file to the folder, where we listed some metadata, like the date and the url of the download, etc. Usually, these files cannot be opened with Presto, as the file format is incompatible in most of the cases.

The next zone is the landing zone. We needed to develop specific scripts, called loader scripts, converting and extracting the raw datasets into this zone. We converted the data to a text file in JSON Lines format (<https://jsonlines.org>), which can be opened by Presto. This is a specific JSON format, where each line of the text file represents a single JSON record. Each JSON file for a given dataset is placed into a separate sub-folder in the landing zone. It is then registered by Presto which sees the dataset now as a table. Then, Presto will be able to load the data, and perform other operations, for example it can execute simple data transformations on the data. However, we do not recommend using the tables in this zone for querying, because processing the large JSON files is very slow.

Using Presto, we converted the data from the landing zone into an optimized (ordered, indexed, binary) format to the next zone, which is the master zone. The main idea is that we use the tables in the master zone later for analytical queries. Here the data is in a more detailed and exact format, called Optimized Row Columnar (ORC), which is a free and open-source column-oriented data storage format (<https://orc.apache.org/docs/>). With ORC, Sherlock can perform SQL queries much more faster than using the JSON text file format from the zone below. If necessary, advanced bucketing or partitioning on these tables can be used to optimize the given queries. Furthermore, the master zone contains the 'single source of truth'. This means that the data here cannot be changed, only extended upon, for example adding a new version of the datasets.

The last zone is the Project zone. This is where we save the tables that are needed only for specific projects. We can even create multiple project zones, one for each group, project or user. It is important to have a rule to indicate the owner of the tables, as mentioned before this dramatically increases the effectiveness of Sherlock and ease of maintenance.

Functionality of Sherlock

Here, we describe the key steps of how the query engine and the Data Lake can be used together (Figure 4). The first step is to submit an SQL query to the Presto query engine, then it will connect to the Data Lake and fetch the necessary data files. The third step is to execute the in-memory distributed query and then it takes back the results to the user.

It is important to mention that Sherlock has been designed for biologists, especially for network and system biologists. It can contain specific, interaction, expression and genome data-related databases thanks to loader scripts that are able to create the specific file formats from the source databases and upload them into the Data Lake. With these individual loader scripts, users can make and work with the necessary file formats from the different data sources without major time and coding efforts with their own scripts. The whole source code of the platform and the loader scripts (Table 1) are freely available on Github: <https://github.com/earlham-sherlock/earlham-sherlock.github.io>

Figure 3. An example strategy, how one can store data inside the Data Lake. The first part of the folder name is the name of the given database/dataset. In that case the human genome. The next part is the version of the data and the last is the uploaded date to the Data Lake.

Table 1. The specific loader scripts, which are already included in Sherlock's github repository.

Datatype	Loader script	Source	Reference
General datasets	DBSNP database	https://www.ncbi.nlm.nih.gov/snp/	(Smigielski et al. 2000)
	Gene Ontology	http://geneontology.org	(Ashburner et al. 2000)
	Gene Ontology Annotations		
	Human Genome	https://www.gencodegenes.org	-
	Uberon Gene Ontology	https://www.ebi.ac.uk/ols/ontologies/uberon	(Mungall et al. 2012)
	Uniprot ID Mapping data	https://www.uniprot.org	(UniProt Consortium 2021)
Interaction databases	BioPlex database	https://bioplex.hms.harvard.edu	(Huttlin et al. 2020)
	Dorothea database	https://dorothea.opentargets.io/#/	(Garcia-Alonso et al. 2018)
	HINT database	http://hint.yulab.org	(Das & Yu 2012)
	HuRI database	http://www.interactome-atlas.org	(Luck et al. 2020)
	InBioMap database	https://inbio-discover.com	(Li et al. 2017)
	IntAct database	https://www.ebi.ac.uk/intact/	(Orchard et al. 2014)
	IRefIndex database	http://irefindex.uio.no	(Razick et al. 2008)
	Mentha database	https://mentha.uniroma2.it	(Calderone et al. 2013)
	Omnipath database	https://omnipathdb.org	(Türei et al.)
	STRING database	https://string-db.org	(Szklarczyk et al. 2019)
Expression data	Bgee database	https://bgee.org	(Bastian et al. 2021)

Use Cases

One of the most crucial and common tasks for those working in the field of bioinformatics is ID mapping. It is difficult to work with many different datasets from many different sources, all carrying diverse identifiers. In addition, these identifiers can have clashing structures, which makes it complicated and time consuming to work with them. Users have to make different scripts or use different tools to work with these identifiers at once. Nowadays the main and key idea behind these ID mapping steps is to have a separate table or tables, called mapping tables, which contain the different identifiers, but only those. The best practice is to have only one mapping table which contains all of the necessary identifiers for a given project. The limitation with this approach is when a team has to work with so many different identifiers at once means that this mapping table can be really large, and it can take a lot of time to go through it and get the data, which is needed.

In Sherlock, users can work with just one mapping table, which can be made easily with the help of the provided loader scripts from the Github repository, and it can significantly shorten the time needed for ID mapping. Sherlock can execute these queries very quickly despite the large size of the mapping tables, owing to the implemented ORC format. To demonstrate this capability in Sherlock, we search for genes from the STRING database (Szkarczyk et al. 2019), which are expressed only in the brain.

Figure 5 shows three tables. The string_proteins table contains the information from the STRING database (ensembl ID, taxonomy ID and the pubmed ID of the article); the mapping table contains the mapping between the different type of protein identifiers from different sources (ensembl (https://www.ensembl.org/info/genome/stable_ids/index.html) and uniprot (https://www.uniprot.org/help/accession_numbers); and the tissues table includes tissue information about the protein (uniprot identifier, tissue identifier and tissue name) (Figure 5).

```
SELECT string_proteins.ens_id,
FROM master.string_proteins
LEFT JOIN master.mapping ON string_proteins.ens_id = mapping.ens_id
LEFT JOIN master.tissues ON mapping.uniprot_id = tissues.uniprot_id
WHERE tissues.bto_name = 'brain' ;
```

The SQL query will connect the three tables through the matching protein identifiers, selecting only those which are expressed in the brain according to the matching identifiers (light green and purple). The query results in the first two proteins from the string_proteins table (light green color). If the script does not find any match between the ensembl and uniprot IDs, it skips them, like in the case of the third protein in the string_proteins table.

Use Case 2: Tissue specificity

In this example, we would like to query the top 100 most highly expressed genes in a given tissue (in this case the human colon), and their protein interactors. The limitation with this use case is similar to the previous one: the user has to download the different structured interaction databases from web resources and write scripts or use online tools to work with the data at once. These are also prerequisite steps with the Sherlock platform (having to download the different sources and running preprocessing steps before working with them), but with the provided loader scripts, the user only has to do it once, and it is less time consuming. With Sherlock, the user can easily get this data from multiple interaction tables at once very quickly, thanks to the ORC format with the following SQL query:

```
SELECT molecule_id, molecule_id_type, tissue_uberoid, tissue_uberoid_name, score,
interactor_a_id, interactor_b_id
FROM master.bgee_2020_11_16 bgee
LEFT JOIN master.omnipath_2020_10_04 ON bgee.molecule_id = omnipath_2020_10_04.-
interactor_a_id
LEFT JOIN master.omnipath_2020_10_04 ON bgee.molecule_id = omnipath_2020_10_04.-
interactor_b_id
WHERE tax_id = 9606
AND tissue_uberoid_id = 1155
ORDER BY score DESC
LIMIT 100;
```

string_proteins			mapping		tissues		
ens_id	tax_id	pubmed_ids	ens_id	uniprot_id	uniprot_id	bto_id	bto_name
ensg11867234247	9606	15328335	ensg11867234247	ph39f7	ph39f7	142	brain
ensg98236453842	9606		ensg98236453842	ph39f7	ph863s	476	foot
ensg36438523943	9606	14847410, 9368760	ensg00034783463	ph863s			

Figure 5. Example query, how can we map between different identifiers with Sherlock. It shows the three tables where we want to map between. All of the three tables are located in the master zone inside the Data Lake.

This query will select the top 100 highly expressed genes from a table imported from the BGee resource (Bastian et al. 2021) to the master zone. The result will be filtered to return only those genes, which are expressed in the colon, and the query will select their protein interactors as well. The results are ordered by the score, so only the colon genes with the top 100 score and their first neighbours will return as the results.

Use Case 3: Network enrichment

In the third example, we would like to enrich a network with interaction data from the Data Lake with the help of Sherlock. We have certain proteins of interest shown on Figure 6, and we would like to investigate if there is any relationship between them (find the interactions). We would also like to enrich this network and our proteins with their first neighbours, using an interaction table loaded from the OmniPath (Türei et al.) (<https://omnipathdb.org>) database.

The SQL query for finding the connections between the proteins (Figure 6A) is the following:

```
SELECT interactor_a_id, interactor_b_id
FROM master.omnipath_2020_10_04
WHERE interactor_a_id IN ('o95786', 'q96eq8', 'q6zsz5', 'q01113')
AND interactor_b_id IN ('o95786', 'q96eq8', 'q6zsz5', 'q01113');
```

To enrich our network with the first neighbours of the given proteins we have to use a different SQL query (Figure 6B), as below:

```
SELECT interactor_a_id, interactor_b_id
FROM master.omnipath_2020_10_04
WHERE interactor_a_id IN ('o95786', 'q96eq8', 'q6zsz5', 'q01113')
OR interactor_b_id IN ('o95786', 'q96eq8', 'q6zsz5', 'q01113');
```

Only a single logical operator changed between the two queries (AND > OR). The reason is, if we want to find the interactions between the proteins, the query has to select only those interactions from the OmniPath table, where the source and the target protein are uniformly included among the proteins we examined. But in the other case, when we want to enrich the network, it is enough to select all of the interactions where either the source or the target proteins are among the proteins of our interest.

Figure 6. Example query, how can we map between different identifiers with Sherlock. A: Here you can see how we can find interconnections between our four chosen proteins. **B:** How can we enrich our network with the first neighbours of the given proteins.

Discussion

Sherlock provides a new, gap-filling method for computational biologists, who want not only to store, but very easily and quickly convert, query, generate or even share biological data. This novel platform provides a simple, user-friendly interface to work with common and widely used big data technologies, such as Docker or PrestoDB. Thanks to the ORC format that Sherlock uses, users can work with extremely large datasets in a relatively short time that can facilitate any given research project.

Since we made Sherlock, plenty of platforms like Sherlock have appeared worldwide, but none of them was designed specifically to the field of biology. One of them for example is the Qubole platform (<https://www.qubole.com>). This platform also offers data storage and analytical queries solutions as well, but the Sherlock platform is cheaper. The data storage place and the virtual machines are also needed to be paid, but all of the source code is freely available and can be customized to the user's liking. On the other hand, Sherlock is specifically designed for biologists. It contains specific database loader scripts which they are able to create and upload the specific file formats to the Data Lake. We are constantly upgrading the already included datasets in our Data Lake. We will provide in the Github repository new database loader scripts (to extend interaction and expression data) and different mapping scripts as well. We will also develop different converting scripts to easily handle between Sherlock compatible and other file formats, such as TSV (tab separated value) or CSV (comma separated value).

Sherlock has a lot of features, which are the followings: 1) store all datasets in redundant and organized cloud storage, 2) convert all datasets to common, optimized file formats, 3) execute analytical queries on top of data files, 4) share datasets among different teams/projects, 5) generate operational datasets for certain services or collaborators, 6) it is really useful for any groups/teams in the field of computational biology, who has to work with very large datasets for their projects.

In the future, we would like to include and upload more and more database loader scripts to cover as many interaction and expression databases as possible. Furthermore we are planning to improve our source code, to make more detailed documentation. Right now, the update of the already included databases in the Data Lake is done manually, but we also would like to automate this with a script, which can handle all of the updates at once. Furthermore, we would like to include more common and general – in the field of biology – examples and show it in the repository. Furthermore, we are planning to develop some tutorials on how Sherlock can be used easily and what information is needed for the given projects. We are going to create short online courses and tutorials as well. Our main goal is to disseminate the Sherlock platform widely and we hope it can be useful and great help to the researchers.

Conclusion

Sherlock provides an open-source platform empowering data management, data analytics and collaboration through modern big data technologies. Utilizing the dockerization of Presto and the Hive Metastore, Sherlock is not only powerful but also a flexible and fast solution to effectively and efficiently store and analyze large biological datasets. Sherlock can be used to execute queries on top of a Data Lake, where all the data is stored in a 'folder-like' structure providing the added benefit of well defined folder structures also helps and encourages correct data management of biological data. Having a scalable query engine and using ORC format, Sherlock can run SQL queries much faster than other solutions, meaning the user can spend more time working with the data than waiting for a search result, which can significantly reduce the lead time of the research project. In conclusion, Sherlock, through the repurposing concepts and open source tools created by large software companies, provides a 'plug and play' state-of-the-art data storage platform for large biological datasets.

Data availability

All data underlying the results are available as part of the article and no additional source data are required.

Software availability

Software available from: <https://earlham-sherlock.github.io/>

Source code available from: <https://github.com/earlham-sherlock/earlham-sherlock.github.io>

Archived source code available from: <http://doi.org/10.5281/zenodo.4738516> (Bohár et al. 2021)

License: MIT

Acknowledgements

We would like to thank all of the members of the Korcsmaros Group for their advice and feedback.

References

- Ashburner M, Ball CA, Blake JA, *et al.*: **Gene Ontology: tool for the unification of biology.** *Nat. Genet.* 2000; **25**(1): 25–29.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
- Bohár B, Szalay-Beko M, Fazekas D, *et al.*: **earlham-sherlock/earlham-sherlock.github.io: First release of the official Sherlock platform (Version v1.0.0).** *Zenodo.* 2021, May 5.
[Publisher Full Text](#)
- Bastian FB, Roux J, Niknejad A, *et al.*: **The Bgee suite: integrated curated expression atlas and comparative transcriptomics in animals.** *Nucleic Acids Res.* 2021; **49**(D1): D831–47.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
- Calderone A, Castagnoli L, Cesareni G: **mentha: a resource for browsing integrated protein-interaction networks.** *Nat. Methods.* 2013; **10**(8): 690–91.
[PubMed Abstract](#) | [Publisher Full Text](#)
- Das J, Yu H: **HINT: High-quality protein interactomes and their applications in understanding human disease.** *BMC Syst. Biol.* 2012; **6**: 92.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
- Dean J, Ghemawat S: **MapReduce: simplified data processing on large clusters.** *Commun. ACM.* 2008; **51**(1): 107.
- Garcia-Alonso L, Iorio F, Matchan A, *et al.*: **Transcription factor activities enhance markers of drug sensitivity in cancer.** *Cancer Res.* 2018; **78**(3): 769–780.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
- Greene CS, Tan J, Ung M, *et al.*: **Big data bioinformatics.** *J. Cell. Physiol.* 2014; **229**(12): 1896–1900.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
- Huttlin EL, Bruckner RJ, Navarrete-Perea J, *et al.*: **Dual Proteome-scale Networks Reveal Cell-specific Remodeling of the Human Interactome.** *BioRxiv.* 2020.
- Kasson PM: **Computational biology in the cloud: methods and new insights from computing at scale.** *Biocomputing 2013.* WORLDSCIENTIFIC; 2012, pp. 451–53.
[PubMed Abstract](#)
- Khine PP, Wang ZS: **Data lake: a new ideology in big data era.** *ITM Web of Conferences.* 2018; **17**: 03025.
[Publisher Full Text](#)
- Li T, Wernersson R, Hansen RB, *et al.*: **A scored human protein-protein interaction network to catalyze genomic interpretation.** *Nat. Methods.* 2017; **14**(1): 61–64.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
- Luck K, Kim D-K, Lambourne L, *et al.*: **A reference map of the human binary protein interactome.** *Nature.* 2020; **580**(7803): 402–8.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
- Marx V: **The Big Challenges of Big Data.** *Nat. Methods.* 2013.
[Publisher Full Text](#)
- Matthias K, Kane SP: *Docker: Up & Running: Shipping Reliable Containers in Production.* Sebastopol, CA: O'Reilly Media. 1st ed. 2015.
- Mungall CJ, Torniai C, Gkoutos GV, *et al.*: **Uberon, an integrative multi-species anatomy ontology.** *Genome Biol.* 2012; **13**(1): R5.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
- Orchard S, Ammari M, Aranda B, *et al.*: **The MIntAct project - IntAct as a common curation platform for 11 molecular interaction databases.** *Nucleic Acids Res.* 2014; **42**(Database issue): D358–63.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
- Razick S, Magklaras G, Donaldson IM: **iRefIndex: a consolidated protein interaction database with provenance.** *BMC Bioinformatics.* 2008; **9**: 405.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
- Silva YN, Almeida I, Queiroz M: **SQL: from traditional databases to big data.** *Proceedings of the 47th ACM Technical Symposium on Computing Science Education - SIGCSE '16.* New York, New York, USA: ACM Press; 2016; pp. 413–18.
- Smigielski EM, Sirotkin K, Ward M, *et al.*: **dbSNP: a database of single nucleotide polymorphisms.** *Nucleic Acids Res.* 2000; **28**(1): 352–55.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
- Smith R: *Docker Orchestration.* 2017; Packt Publishing
- Szklarczyk D, Gable AL, Lyon D, *et al.*: **STRING v11: protein-protein association networks with increased coverage, supporting functional discovery in genome-wide experimental datasets.** *Nucleic Acids Res.* 2019; **47**(D1): D607–13.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
- Türei D, Valdeolivas A, Gul L, *et al.*: **Integrated intra- and intercellular signaling knowledge for multicellular omics analysis.** *Mol. Syst. Biol.* 2017; **13**(3): e9923.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
- UniProt Consortium: **UniProt: the universal protein knowledgebase in 2021.** *Nucleic Acids Res.* 2021; **49**(D1): D480–89.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)

Open Peer Review

Current Peer Review Status: ? ?

Version 1

Reviewer Report 29 June 2021

<https://doi.org/10.5256/f1000research.56112.r85970>

© 2021 Doncheva N. This is an open access peer review report distributed under the terms of the [Creative Commons Attribution License](#), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Nadezhda T. Doncheva

Novo Nordisk Foundation Center for Protein Research, University of Copenhagen, Copenhagen, Denmark

In this article, the authors present an open-source platform called Sherlock, which can be very useful for computational biologists working with a lot of different datasets within one or several projects. The platform is designed to make use of well-known resources and concepts such as Docker, and PrestoDB. Sherlock is easily extendable and can be used both on a local computer or in connection with a cloud storage solution. Once Sherlock is properly set up and the needed data sets imported and formatted, the data can be integrated and analyzed using SQL queries as shown in the three simple use cases.

Sherlock can become a great addition to the suite of tools used by computational biologists, especially when it comes to bigger projects involving a lot of public datasets as well as own data generated by collaborators. The article is nicely written and gives a good overview of the Sherlock platform as well as possible use cases. However, it can be further improved in three main aspects as detailed below: making clear who is the target audience, improving the documentation about setting up Sherlock, and improving the flow of the Implementation and Operation section.

Both the title and abstract mention that Sherlock is designed for biologists and only on page 6, the text mentions computational biologists specifically. However, both from the description of the setup and the specific use cases, it seems to me that the actual target audience should be clearly stated as “computational biologists”. It is worth considering changing the “for biology” part in the title since it is not very clear or specific as it is now.

One major concern is that neither the manuscript nor the online documentation seem to describe in enough detail how to set up the data needed for the three presented use cases. There is a nice deployment guide and the use case examples seem reasonable, but the step in-between is missing. The use cases assume that the user has an already configured Sherlock system with the necessary files stored, properly formatted and the right tables created, etc. I specifically checked the GitHub documentation, among others the “Loading ... data to Sherlock” sections and I don't

think enough detail is given for a user like me to set it up (and I consider myself a computational biologist). I would need to know which raw data files from STRING were retrieved and stored in the raw zone, how were the JSON files generated that need to be copied to the landing zone, etc. I was able to find the scripts, which should help for this step (<https://github.com/earlham-sherlock/earlham-sherlock.github.io/tree/master/loaders>) but they are also not described in enough detail, especially in connection with the use cases. So, what I think is needed is the following:

1. add a short description to each use case in the manuscript about which databases are needed to execute the example queries;
2. on GitHub, add a step-by-step tutorial for how to set up the specific databases/tables needed for the use cases. For use case 1, the tutorial should include information about how and from what data to create the tables "master.mapping", "master.tissues", "master.string_proteins", while for use case 2 and 3, the needed tables are "master.bgee_2020_11_16" and "master.omnipath_2020_10_04".

Another major comment is that the structure of the whole section "Implementation and Operation" can benefit from a bit of reorganization and section renaming. In the following, I will give some specific details and examples of how this can be done.

- The Overview section begins with a lengthy description of Docker and Docker Swarm and only gives an overview of the different components of Sherlock at the end of the section. The section will read much better if it begins with a high-level overview of the different components as it is partly done in the third paragraph (last one on page 3). This part can also refer to Figure 1. Then, each of the components can be explained in more or less detail (depending on the order and what follows in the next sections). For example, the end of paragraph one, which describes the scalability of Docker Swarm, also fits well with the last paragraph of the Overview section. This section could also be renamed to "Overview of the Sherlock platform".
- In the Overview section, there seems to be a bit of confusion about the architecture & setup versus the usage workflow. The second paragraph starts with a sentence stating that "the first step of using Sherlock is to start a Docker Swarm", while the third paragraph states that "the starting step for using Sherlock is to submit an SQL query to the Presto query engine". I can see how both of these statements are true, but it will be easier to read if it is explained that one thing is part of the general, one-time setup of the system (which also fits more with the whole architecture), while the other one is the actual user workflow in a day-to-day usage (more relevant to the "Functionality of Sherlock" section).
- As a follow-up on that, it would be possible to move the whole description of Docker together with the "Query engine" and "Hive metastore" subsections into a section entitled "Architecture" or optionally "Architecture and setup" (instead of "Sherlock as a platform").
- The last section about the functionality of Sherlock is currently very short and actually describes this day-to-day workflow. It could easily become part of the first overview section or else be extended by a few more details that do not fit so well into the Overview/Architecture sections.
- So, for the whole big section, the order would be "Overview of the Sherlock platform",

“Architecture (and setup)”, “Deployment”, “The Data lake – data storage” and optionally “Functionality of Sherlock”.

In the following list, I have pointed out several minor things about the text or figures that can also be improved.

- The abstract says that “Sherlock is designed to analyse, process, query and extract the information from extremely complex and large datasets.” I think to be precise, it should say that it is designed “to facilitate/enable/allow users in analysing, processing, ... ”
- The description of Docker in the first paragraph of the Overview section could be shortened a bit (or moved to a more fitting place a bit later in the description).
- The legend of Figure 1 can be improved by describing the elements from left to right instead of right to left. It is not clear what a “Minio S3 server” is since it is not mentioned anywhere else in the manuscript. The first column of boxes in the Figure could also be surrounded by a bigger box and referred to as the user input (configuration) or something similar. Then it would be easier to refer to it.
- Page 4, second paragraph: “A Query Engines... are distributed” should be either singular or plural but not both.
- Page 4, third paragraph: the phrase “designed for working data” should probably be “designed for working with data” and “which held in” should be “which is held in”.
- Page 4, third paragraph: the sentence “Moreover it is particularly really useful to handle with structured data” can be rephrased for more clarity.
- Page 4, last paragraph: the sentence “With regards to Sherlock, Presto stores the metadata about the folders in the Data Lake and Hive metastore, which contains only the meta information about the different folders, which is in the Data Lake” also needs to be rephrased. There are too many “which” and it is not clear what refers to what and what information is actually contained/known to Presto and Hive metastore.
- Figure 2 does not look very nice in the PDF as opposed to the online version on GitHub, which has nicer formatting and quality. I would recommend replacing the figure in the PDF with the online version and making sure it still looks fine.
- Page 5, last paragraph: “on a local machine, on the cloud and the data...” should probably be “on a local machine and on the cloud and that the data...”.
- Page 5, last paragraph: the semicolon in the sentence “the most common Data Lake solutions;” could be replaced by a colon and “and with the Simple...” by “and Simple ...”.
- Page 6, first paragraph of “Functionality of Sherlock”: how should the last sentence be understood? Does the user execute a “in-memory distributed query” or is this done also by the Presto engine?
- Page 6, last paragraph: the comma in “contain specific, interaction, ...” is unnecessary.

- Page 7, there is an issue with the Omnipath reference in the table, it looks different and leads to a log-in page in the online version of the article.
- Page 7, first sentence "...we will outline three different use cases on how Sherlock can be used and what we can use it for" could be rephrased to sound better. Maybe just "... we will outline three different use cases of Sherlock".
- Page 9, second paragraph: should this sentence "to enrich a network with interaction data" rather say "to retrieve interaction data for a gene list"?
- Figure 6: The two actions "Enrich with interconnections" and "Enrich with first neighbors" are repeated. I think it is enough to state this once below the arrow since it nicely describes the actual process. The picture before and after the arrow could have some other labels, but it is also fine if they do not have their own label.
- Page 10, second paragraph: the sentence "it contains specific database loader scripts which they are able to create and upload specific file formats to the Data Lake" needs some revising to make clear who is "they". Maybe it should say "which the users are able to use to upload or can create themselves...".
- Page 10, third paragraph: "followings" should be "following".
- Page 10, last paragraph: there is probably a stop (sentence end) missing between "structure" and "providing" in "a 'folder-like' structure providing the added...".
- I am not able to see the images in the html version of the article in Firefox on Mac, I can only see a name (f1db3fb0-55ef-43b8-9ea8-7f6486a79c9a_figure1.gif). If I right-click and try to see it in another tab, I see some XML code that says Access denied. I checked and I don't have the same issue with other articles.

Is the rationale for developing the new software tool clearly explained?

Partly

Is the description of the software tool technically sound?

Yes

Are sufficient details of the code, methods and analysis (if applicable) provided to allow replication of the software development and its use by others?

Partly

Is sufficient information provided to allow interpretation of the expected output datasets and any results generated using the tool?

Yes

Are the conclusions about the tool and its performance adequately supported by the findings presented in the article?

Yes

Competing Interests: No competing interests were disclosed.

Reviewer Expertise: Computational and network biology, development of software tools, analysis and integration of large datasets

I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard, however I have significant reservations, as outlined above.

Author Response 21 Jul 2022

Tamás Korcsmáros

Response to Reviewers for Manuscript ID: f1000research.52791.1

We thank the reviewers for their feedback on our manuscript and the constructive comments. The comments and questions led to a significant and appropriate revision of the manuscript.

Answers for reviewer Nadezhda T. Doncheva

In this article, the authors present an open-source platform called Sherlock, which can be very useful for computational biologists working with a lot of different datasets within one or several projects. The platform is designed to make use of well-known resources and concepts such as Docker, and PrestoDB. Sherlock is easily extendable and can be used both on a local computer or in connection with a cloud storage solution. Once Sherlock is properly set up and the needed data sets imported and formatted, the data can be integrated and analyzed using SQL queries as shown in the three simple use cases.

Sherlock can become a great addition to the suite of tools used by computational biologists, especially when it comes to bigger projects involving a lot of public datasets as well as own data generated by collaborators. The article is nicely written and gives a good overview of the Sherlock platform as well as possible use cases. However, it can be further improved in three main aspects as detailed below: making clear who is the target audience, improving the documentation about setting up Sherlock, and improving the flow of the Implementation and Operation section.

We thank the Reviewer for these general comments and the summary about the manuscript. We also really appreciate the suggestions about the three main aspects.

Both the title and abstract mention that Sherlock is designed for biologists and only on page 6, the text mentions computational biologists specifically. However, both from the description of the setup and the specific use cases, it seems to me that the actual target audience should be clearly stated as "computational biologists". It is worth considering changing the "for biology" part in the title since it is not very clear or specific as it is now.

Thank you very much for raising it, it is a good point. We agree with the Reviewer's view, and modified the manuscript throughout to reflect that Sherlock is primarily for

computational biologists.

One major concern is that neither the manuscript nor the online documentation seem to describe in enough detail how to set up the data needed for the three presented use cases. There is a nice deployment guide and the use case examples seem reasonable, but the step in-between is missing. The use cases assume that the user has an already configured Sherlock system with the necessary files stored, properly formatted and the right tables created, etc. I specifically checked the GitHub documentation, among others the "Loading ... data to Sherlock" sections and I don't think enough detail is given for a user like me to set it up (and I consider myself a computational biologist). I would need to know which raw data files from STRING were retrieved and stored in the raw zone, how were the JSON files generated that need to be copied to the landing zone, etc. I was able to find the scripts, which should help for this step (<https://github.com/earlham-sherlock/earlham-sherlock.github.io/tree/master/loaders>) but they are also not described in enough detail, especially in connection with the use cases. So, what I think is needed is the following:

add a short description to each use case in the manuscript about which databases are needed to execute the example queries.

On GitHub, add a step-by-step tutorial for how to set up the specific databases/tables needed for the use cases. For use case 1, the tutorial should include information about how and from what data to create the tables "master.mapping", "master.tissues", "master.string_proteins", while for use case 2 and 3, the needed tables are "master.bgee_2020_11_16" and "master.omnipath_2020_10_04".

Thank you very much for your advice, these are really good and useful points. We have added the suggested clarifications for the github repository and for the Use Cases section in the revised manuscript as well.

Another major comment is that the structure of the whole section "Implementation and Operation" can benefit from a bit of reorganization and section renaming. In the following, I will give some specific details and examples of how this can be done.

The Overview section begins with a lengthy description of Docker and Docker Swarm and only gives an overview of the different components of Sherlock at the end of the section. The section will read much better if it begins with a high-level overview of the different components as it is partly done in the third paragraph (last one on page 3). This part can also refer to Figure 1. Then, each of the components can be explained in more or less detail (depending on the order and what follows in the next sections). For example, the end of paragraph one, which describes the scalability of Docker Swarm, also fits well with the last paragraph of the Overview section. This section could also be renamed to "Overview of the Sherlock platform".

Thank you for bringing this to our attention, we have now improved this section and made the suggested restructuring changes.

In the Overview section, there seems to be a bit of confusion about the architecture & setup versus the usage workflow. The second paragraph starts with a sentence stating that "the first step of using Sherlock is to start a Docker Swarm", while the third paragraph states that "the starting step for using Sherlock is to submit an SQL query to the Presto query engine". I can see how both of these statements are true, but it will be easier to read if it is explained that one thing

is part of the general, one-time setup of the system (which also fits more with the whole architecture), while the other one is the actual user workflow in a day-to-day usage (more relevant to the "Functionality of Sherlock" section).

Thank you for raising it as well, we have fixed these inaccuracies. In the Use Cases section, we added a general set up guide of the platform and the Data Lake. After that, we introduce how to run basic analytical SQL queries in different use cases.

As a follow-up on that, it would be possible to move the whole description of Docker together with the "Query engine" and "Hive metastore" subsections into a section entitled "Architecture" or optionally "Architecture and setup" (instead of "Sherlock as a platform").

Thank you for your suggestion. We changed the section names and structure accordingly.

The last section about the functionality of Sherlock is currently very short and actually describes this day-to-day workflow. It could easily become part of the first overview section or else be extended by a few more details that do not fit so well into the Overview/Architecture sections.

Thank you, this is a good idea. We created a new subsection for the whole docker part and also extended it with more details.

So, for the whole big section, the order would be "Overview of the Sherlock platform", "Architecture (and setup)", "Deployment", "The Data lake – data storage" and optionally "Functionality of Sherlock".

Thank you for your advice. We have restructured the whole Implementation and Operation section based on your suggestions, and also we fixed the minor points that you raised.

Thank you very much for all of the constructive comments that helped us to improve the manuscript.

Competing Interests: No competing interests

Reviewer Report 17 June 2021

<https://doi.org/10.5256/f1000research.56112.r85971>

© 2021 Morris J. This is an open access peer review report distributed under the terms of the [Creative Commons Attribution License](#), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

John H. Morris

Resource for Biocomputing, Visualization and Informatics, Department of Pharmaceutical Chemistry, University of California, San Francisco, San Francisco, CA, USA

In this article, the authors describe a system, Sherlock, for storing and accessing data using SQL. The system described is flexible, and uses modern approaches to the standard ETL (extract, transform, load) processes common in a data warehouse solution. The system seems like it would be extremely useful for a narrow set of use cases, somewhat useful for a larger set of use cases, and of questionable utility for most computational biology use cases.

Sherlock's main utility, it seems to me, is for facilities that have large repositories of self-generated data. In that circumstance, the overhead of learning the column names, appropriate foreign keys, and the appropriate SQL syntax to manipulate the data would be well-warranted, and I would imagine in those circumstances, the data could be stored locally on some fast media that would provide significant performance advantages, particularly given the distributed nature of presto. I can also see Sherlock being useful in circumstances where there is a relatively static pipeline that operates on data sources that are more commonly updated. In this circumstance, the ability to have scripts to stage the data and use a common SQL syntax that doesn't change would be useful. In other circumstances, I'm less convinced. First, while I appreciate the value of the multiple-level folder structure (Raw Zone --> Landing Zone --> Master Zone --> Project Zone) that results in a number of copies of databases that could be extremely large. The examples given are relatively small, but consider downloading TrEMBL, which is 132 Gb compressed, and reprocessing that to get at least 3 copies (assuming one would subset it for the Project Zone). The scary part is that TrEMBL is small by modern standards -- image datasets are significantly larger. Second, while I agree that for software developers, SQL is a well-known and convenient query language, it is certainly non-trivial to construct efficient queries and not the best interactive tool for exploring data sets. Further, there is no attempt to fit this tool into the more broad set of tools commonly used by computational biologists. For example, why would I use Sherlock for scRNA-seq analysis if I have access to Galaxy or AnVIL?

So, not to be totally negative, I think Sherlock has a place in the tool suite for computational biology, but I don't think that place is well articulated in the article. My recommendations for improving the article are:

1. The second sentence in the introduction talks about working on bespoke datasets, which I suspect is the crux of Sherlock. The rest of the article seems to ignore that, and even the use cases focus purely on public data. This weakens the case for Sherlock. The use cases are trivially (OK, so relatively) easy using a combination of python, perhaps with pandas, data files and web services. The power of Sherlock is the ability to *integrate* public databases with bespoke databases, but that really isn't highlighted sufficiently. That needs to be the focus on the user cases, I think.
2. Discuss Sherlock in the context of other tools (not technologies) that support computational biology, such as Galaxy and AnVIL as two examples. When would I use those tools? When would I use Sherlock? Why is Sherlock better in that case?
3. Be clearer as to the target user. In several cases, the article states that this is "designed specifically for biologists". I actually don't think that's correct. It's designed for computational biologists or bioinformaticians with a very strong computational background. I don't know many bench biologists who would be able to formulate some of the SQL queries, even if it was easy to figure out all of the column names. I don't *think* the

authors believe that SQL is the user interface for the system as much as an API for accessing the data, but that also wasn't clear in the text.

4. The role of Hive is somewhat obscure to me. I got that it stores the metadata, but **what** metadata? Is that where I can look up the schema to get the column names for joins and projections? In the use cases, it seems like there were several jumps in logic. Where does the user figure out that the ensemble ID in the STRING database is stored as "ens_id"? How well exposed is the schema for each of the tables in the data lake and how does a user (who didn't happen to be the one who imported the data) figure that out?
5. One of the main points in the article is the potential for high performance due to the parallelism of presto, but there were no performance numbers offered, either for the ETL phase or for the queries themselves.
6. Finally, as a more minor point, there are several places where the manuscript could use a bit of an edit pass. For example, on page 2 it states "A Query Engines, for example " shouldn't be plural.

Sherlock clearly represents a **lot** of work, and I do believe that it could be valuable to the appropriate audience and that it should be published. My take on the manuscript is that it's not clear who it is written for. If it's written for a computer scientist, then information about performance metrics and local vs. cloud storage trade-offs and scalability, etc., are missing. If it's written for the computational biologist, then it's not really necessary to address the trade-offs between docker swarm and kubernetes, or why SQL can be considered composed of three main sub-languages, but it is important to explain why Sherlock is better than Galaxy or AnVIL and a little more detail on how the user figures out things like column names, etc.

Sorry if this comes across as overly negative. I honestly do see how much work is involved and I can imagine where it might be extremely useful in some environments, but that just doesn't come across clearly in this submission.

Is the rationale for developing the new software tool clearly explained?

Partly

Is the description of the software tool technically sound?

Yes

Are sufficient details of the code, methods and analysis (if applicable) provided to allow replication of the software development and its use by others?

Yes

Is sufficient information provided to allow interpretation of the expected output datasets and any results generated using the tool?

Yes

Are the conclusions about the tool and its performance adequately supported by the findings presented in the article?

Partly

Competing Interests: No competing interests were disclosed.

Reviewer Expertise: computational biology

I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard, however I have significant reservations, as outlined above.

Author Response 21 Jul 2022

Tamás Korcsmáros

Response to Reviewers for Manuscript ID: f1000research.52791.1

We thank the reviewers for their feedback on our manuscript and the constructive comments. The comments and questions led to a significant and appropriate revision of the manuscript.

Answers for reviewer John H. Morris

In this article, the authors describe a system, Sherlock, for storing and accessing data using SQL. The system described is flexible, and uses modern approaches to the standard ETL (extract, transform, load) processes common in a data warehouse solution. The system seems like it would be extremely useful for a narrow set of use cases, somewhat useful for a larger set of use cases, and of questionable utility for most computational biology use cases.

Sherlock's main utility, it seems to me, is for facilities that have large repositories of self-generated data. In that circumstance, the overhead of learning the column names, appropriate foreign keys, and the appropriate SQL syntax to manipulate the data would be well-warranted, and I would imagine in those circumstances, the data could be stored locally on some fast media that would provide significant performance advantages, particularly given the distributed nature of presto. I can also see Sherlock being useful in circumstances where there is a relatively static pipeline that operates on data sources that are more commonly updated. In this circumstance, the ability to have scripts to stage the data and use a common SQL syntax that doesn't change would be useful.

We thank the Reviewer for these general comments and we agree with the specific use cases described by the Reviewer on where Sherlock could be useful.

In other circumstances, I'm less convinced. First, while I appreciate the value of the multiple-level folder structure (Raw Zone --> Landing Zone --> Master Zone --> Project Zone) that results in a number of copies of databases that could be extremely large. The examples given are relatively small, but consider downloading TrEMBL, which is 132 Gb compressed, and reprocessing that to get at least 3 copies (assuming one would subset it for the Project Zone). The scary part is that TrEMBL is small by modern standards -- image datasets are significantly larger. Second, while I agree that for software developers, SQL is a well-known and convenient query language, it is certainly non-trivial to construct efficient queries and not the best interactive tool for exploring

data sets. Further, there is no attempt to fit this tool into the more broad set of tools commonly used by computational biologists. For example, why would I use Sherlock for scRNA-seq analysis if I have access to Galaxy or AnVIL?

We thank the Reviewer for this comment, which is really helpful as these were not properly covered in the original submission. Regarding the multi-level folder structure, we added a paragraph into the end of "The Data Lake - data storage" section to explain its usefulness further. Regarding the last part of this comment, we added paragraphs to the "Discussion" section of the revised manuscript.

So, not to be totally negative, I think Sherlock has a place in the tool suite for computational biology, but I don't think that place is well articulated in the article. My recommendations for improving the article are:

The second sentence in the introduction talks about working on bespoke datasets, which I suspect is the crux of Sherlock. The rest of the article seems to ignore that, and even the use cases focus purely on public data. This weakens the case for Sherlock. The use cases are trivially (OK, so relatively) easy using a combination of python, perhaps with pandas, data files and web services. The power of Sherlock is the ability to integrate public databases with bespoke databases, but that really isn't highlighted sufficiently. That needs to be the focus on the user cases, I think.

We would like to thank you for this comment and also thanks for the suggestions, we modified the manuscript accordingly. We added that part into the "Overview of the platform" section.

Discuss Sherlock in the context of other tools (not technologies) that support computational biology, such as Galaxy and AnVIL as two examples. When would I use those tools? When would I use Sherlock? Why is Sherlock better in that case?

Thank you for these important questions not yet addressed in the manuscript. We added new paragraphs about AnVIL and Galaxy into the "Discussion" section to address these.

*Be clearer as to the target user. In several cases, the article states that this is "designed specifically for biologists". I actually don't think that's correct. It's designed for computational biologists or bioinformaticians with a very strong computational background. I don't know many bench biologists who would be able to formulate some of the SQL queries, even if it was easy to figure out all of the column names. I don't *think* the authors believe that SQL is the user interface for the system as much as an API for accessing the data, but that also wasn't clear in the text.*

Thank you for this comment as well. This is a valid point as indeed Sherlock was designed for computational biologists or data scientists. Therefore, we clarified it in the revised manuscript and updated the title as well.

*The role of Hive is somewhat obscure to me. I got that it stores the metadata, but *what* metadata? Is that where I can look up the schema to get the column names for joins and projections? In the use cases, it seems like there were several jumps in logic. Where does the user*

figure out that the ensemble ID in the STRING database is stored as "ens_id"? How well exposed is the schema for each of the tables in the data lake and how does a user (who didn't happen to be the one who imported the data) figure that out?

Thank you for raising this. Hive is actually more of a technical part, which is used by PrestoDB. Sherlock stores metadata (columns structure) about the tables for Presto to query the files. For best practice, each user/user group needs to add a schema to describe what is inside the tables (wiki page, loader scripts, SQL editors). We have extended the manuscript to clarify these points in the "Functionality of Sherlock" section.

One of the main points in the article is the potential for high performance due to the parallelism of presto, but there were no performance numbers offered, either for the ETL phase or for the queries themselves.

Thank you for pointing out that this comparison was missing. We added test measurement to the manuscript into the "The Data Lake - data storage" section.

Finally, as a more minor point, there are several places where the manuscript could use a bit of an edit pass. For example, on page 2 it states "A Query Engines, for example " shouldn't be plural.

Thank you for raising this, we reviewed and corrected the text now.

*Sherlock clearly represents a *lot* of work, and I do believe that it could be valuable to the appropriate audience and that it should be published. My take on the manuscript is that it's not clear who it is written for. If it's written for a computer scientist, then information about performance metrics and local vs. cloud storage trade-offs and scalability, etc., are missing. If it's written for the computational biologist, then it's not really necessary to address the trade-offs between docker swarm and kubernetes, or why SQL can be considered composed of three main sub-languages, but it is important to explain why Sherlock is better than Galaxy or AnVIL and a little more detail on how the user figures out things like column names, etc.*

Thank you very much for this summary and the suggestions. We agree with the Reviewer's view. Accordingly, we modified the manuscript throughout to meet with the interest and needs of a computational biologist. In some cases, we kept some background information (such as trade-offs between docker swarm and kubernetes) but rephrased them in a restructured arrangement.

Sorry if this comes across as overly negative. I honestly do see how much work is involved and I can imagine where it might be extremely useful in some environments, but that just doesn't come across clearly in this submission.

Thank you very much for all of the constructive criticism and helping us to improve the manuscript.

Competing Interests: No competing interests

The benefits of publishing with F1000Research:

- Your article is published within days, with no editorial bias
- You can publish traditional articles, null/negative results, case reports, data notes and more
- The peer review process is transparent and collaborative
- Your article is indexed in PubMed after passing peer review
- Dedicated customer support at every stage

For pre-submission enquiries, contact research@f1000.com

F1000Research