









SOFTWARE TOOL ARTICLE

isa4j: a scalable Java library for creating ISA-Tab metadata

[version 1; peer review: 2 approved]

Dennis Psaroudakis ^{1,2}, Feng Liu¹, Patrick König ¹, Uwe Scholz ¹,
Astrid Junker ¹, Matthias Lange ¹, Daniel Arend ¹

¹Leibniz Institute of Plant Genetics and Crop Plant Research (IPK) Gatersleben, Seeland, 06466, Germany

²Hochschule Mittweida, University of Applied Sciences, Mittweida, 09648, Germany

V1 First published: 03 Dec 2020, 9(ELIXIR):1388
<https://doi.org/10.12688/f1000research.27188.1>

Latest published: 03 Dec 2020, 9(ELIXIR):1388
<https://doi.org/10.12688/f1000research.27188.1>

Abstract

Experimental data is only useful to other researchers if it is findable, accessible, interoperable, and reusable (FAIR). The ISA-Tab framework enables scientists to publish metadata about their experiments in a plain text, machine-readable format that aims to confer that interoperability and reusability. A Python software package (isatools) is currently being developed to programmatically produce these metadata files. For Java-based environments, there is no equivalent solution yet. While the isatools package provides a lot of flexibility and a wealth of different features for the Python ecosystem, a package for JVM-based applications might offer the speed and scalability needed for writing very large ISA-Tab files, making the ISA framework available in an even wider range of situations and environments. Here we present a light-weight and scalable Java library (isa4j) for generating metadata files in the ISA-Tab format, which elegantly integrates into existing JVM applications and especially shines at generating very large files. It is modeled after the ISA core specifications and designed in keeping with isatools conventions, making it consistent and intuitive to use for the community. isa4j is implemented in Java (JDK11+) and freely available under the terms of the MIT license from the Central Maven Repository (<https://mvnrepository.com/artifact/de.ipk-gatersleben/isa4j>). The source code, detailed documentation, usage examples and performance evaluations can be found at <https://github.com/IPK-BIT/isa4j>.

Keywords

ISA-Tab, FAIR data, reproducible research, metadata, Java, object-oriented programming, framework

Open Peer Review

Reviewer Status  

Invited Reviewers

1

2

version 1



03 Dec 2020



report



report

1. **Massimiliano Izzo** , University of Oxford, Oxford, UK
2. **Nils Hoffmann** , Bielefeld University, Bielefeld, Germany

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



This article is included in the [ELIXIR gateway](#).

Corresponding author: Daniel Arend (arendd@ipk-gatersleben.de)

Author roles: **Psaroudakis D:** Conceptualization, Investigation, Software, Writing – Original Draft Preparation, Writing – Review & Editing; **Liu F:** Conceptualization, Investigation, Software, Writing – Review & Editing; **König P:** Conceptualization, Writing – Original Draft Preparation, Writing – Review & Editing; **Scholz U:** Funding Acquisition, Project Administration, Writing – Review & Editing; **Junker A:** Funding Acquisition, Project Administration, Writing – Review & Editing; **Lange M:** Conceptualization, Writing – Review & Editing; **Arend D:** Conceptualization, Investigation, Software, Supervision, Writing – Original Draft Preparation, Writing – Review & Editing

Competing interests: No competing interests were disclosed.

Grant information: This work was supported by the German Ministry of Education and Research (BMBF) through the grants FKZ 031A053B 'DPPN' (assigned to Matthias Lange, Uwe Scholz, and Astrid Junker), FKZ 031A536A 'de.NBI' (assigned to Matthias Lange and Uwe Scholz) and supported by ELIXIR.

Copyright: © 2020 Psaroudakis D *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: Psaroudakis D, Liu F, König P *et al.* **isa4j: a scalable Java library for creating ISA-Tab metadata [version 1; peer review: 2 approved]** F1000Research 2020, 9(ELIXIR):1388 <https://doi.org/10.12688/f1000research.27188.1>

First published: 03 Dec 2020, 9(ELIXIR):1388 <https://doi.org/10.12688/f1000research.27188.1>

Introduction

In recent years, the question of how to publish research data has increasingly come into the limelight of discussions among scholars, funders, and publishers¹. Wilkinson *et al.*² establish a set of principles to ensure that data are shared in a way that is useful to the community and worthwhile for data producers: Data should be findable, accessible, interoperable, and reusable (FAIR) – not only by humans but also by computers. In some scientific fields, there are well-curated, consistent, and strongly integrated databases that provide easy access for both humans and machines, such as Genbank and UniProt for nucleotide and protein sequences^{3,4}. Other areas, like plant phenotyping, do not yet have central databases or established file formats and things become especially difficult when data from different domains need to be published in conjunction. The Investigation-Study-Assay (ISA) framework and the corresponding ISA-Tab file format⁵ provide a clearly defined, machine-readable, and extensible structure for explanatory metadata that bundles common elements while keeping data in separate files using appropriate formats. Several communities have already created specific standards (such as MIAPPE⁶ or MIAME⁷) and infrastructure⁸ based on the ISA framework. Furthermore, tools have been developed for validating, converting, and manually crafting ISA-Tab metadata^{7,9,10}. However, given the ever-increasing volume of research data generated in high-throughput experiments, the manual creation of metadata is simply not feasible in many situations. A Python package called isatools for programmatically generating ISA-Tab metadata is currently under development (<https://isatools.readthedocs.io>) featuring methods to parse, validate, build, and convert ISA files. It also offers a feature to create sample collection and assay run templates according to a specified experimental design which can be useful when planning an experiment. Building ISA-Tab files, isatools provides great flexibility and ease of use: users can create and connect ISA objects in arbitrary order and degree of detail and isatools automatically determines the appropriate formatting when the ISA-Tab text is rendered.

Naturally, this flexibility requires isatools to keep the whole object structure in memory and resolve the optimal path through the object chain when the content is serialized. This can notably impact performance when describing large and complex studies including a high number of replicates and attributes, as for instance required by the MIAPPE standard for plant phenotyping experiments. This could make it challenging to use isatools in interactive and time-sensitive applications. Additionally, in the majority of cases, the desired file structure is already clear beforehand based on such community standards or your own decision of what needs to be documented, so this flexibility is often not needed. We therefore set out to develop a solution that focuses on high performance and scalability, and which would integrate well into JVM-based data publishing ecosystems. The library, called isa4j, addresses these goals by providing interfaces for exporting ISA-formatted metadata not only to files, but also to any data stream provided by the application (e.g. a HTTP response stream in a web application) and using an iterative approach for creating ISA-Tab files: Instead of loading all records into memory and writing them in one go, an output stream is opened, a single record is created, flushed out into

the stream, and then immediately dropped again from memory. This guarantees memory usage to remain constant so that isa4j imposes no limit on the size of the generated metadata and is able to process datasets too big to fit into memory. The output stream can also be picked up by the application and piped into further processing steps, such as calculating checksums or compressing the ISA-Tab content. In exchange, the user needs to structure rows consistently as headers cannot be modified once they are written. The schema in [Figure 1](#) shows the exemplary integration of isa4j into different application scenarios for supporting the FAIR data sharing paradigm. In this article, we explain how isa4j can be used to generate ISA-Tab metadata and compare it to isatools in performance and scalability regarding both quantity and complexity of ISA-Tab entries.

Methods

Implementation

isa4j is implemented in Java (JDK11+) and can therefore also be used with other JVM-based languages like Groovy or Kotlin. It uses the Gradle Build Tool (<https://gradle.org>) to resolve

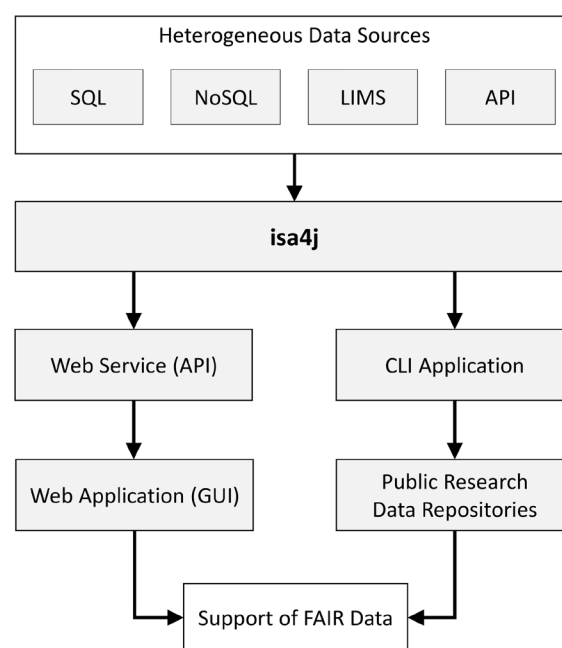


Figure 1. Exemplary integration of isa4j into different application scenarios for supporting the FAIR data sharing paradigm. Heterogeneous data sources like SQL and NoSQL databases, laboratory information management systems (LIMS) and application programming interfaces (API) that store data and metadata of scientific experiments can be fed into isa4j to integrate and transform this data to output complying with the ISA specifications: The isa4j library can, for example, be embedded in command line interface (CLI) applications to create ISA-Tab files in a batch processing manner. It may also be embedded in web services to create ISA-Tab files on the fly via an API based on specific user requirements. ISA-Tab files created with CLI applications could be uploaded to public research data repositories for long-term storage and web applications as graphical user interfaces would allow low-barrier interactive access to experimental data. Both examples demonstrate how isa4j can be used for FAIR data sharing.

dependencies and create arti-facts. Logging is realized via the framework-agnostic SLF4J library (<http://www.slf4j.org/>) so that isa4j works with a variety of logging libraries. The object-oriented Java class structure is modelled according to the published ISA specifications (<https://isa-specs.readthedocs.io>) to make isa4j intuitive to use and keep consistency with other ISA applications. The `Ontology` and `OntologyAnnotation` classes allow linking characteristics, units, and other metadata to established vocabularies such as those collected by the OBO Foundry¹¹.

Operation

isa4j is not an application itself but a software library providing methods for generating ISA-Tab metadata in JVM-based applications or scripts. As a result, operation requires at least a basic level of coding skills in Java or another JVM-based language. When using a build tool like Maven or Gradle, isa4j can simply be added as a dependency to be downloaded from the Central Maven Repository (<https://mvnrepository.com/artifact/de.ipk-gatersleben/isa4j>). Otherwise, the JAR file can be downloaded from there and manually included in the class path. To use isa4j's logging feature, one of the SLF4J bindings needs to be included the same way (<http://www.slf4j.org/manual.html>).

You can then import isa4j classes and start building Investigation, Study, and Assay files. For examples and details on the code interface itself, please consult the current project page (<https://github.com/IPK-BIT/isa4j>) as things may change in future versions and we do not want to confuse you with potentially outdated information.

Scalability evaluation

Scalability of isa4j was assessed and compared to the Python isatools API in two dimensions: number of entries and complexity of entries.

At the simplest complexity level (*Minimal*), Study file rows consisted only of a Source connected to a Sample through a Process, and that Sample connected to a DataFile through another Process in the Assay File, with no Characteristics, Comments, or other additional information (6 columns in total). At the second degree of complexity (*Reduced*), a Characteristic was added to the Sample in the Study File, and the Assay File was expanded to include an intermediary Material Object (11 columns). The third and final level of complexity (*Real World*) was modelled after the MIAPPE v1.1 compliant real-world metadata published for a plant phenotyping experiment (<https://doi.org/10.5447/IPK/2020/3>, 119 columns). Exemplary ISA-Tab output for each of the three complexity levels can be found at <https://ipk-bit.github.io/isa4j/scalability-evaluation.html#complexity-levels>.

For each complexity level, CPU execution time was measured for writing a number of n rows in Study and Assay File each, starting at 1 and increasing in multiplicative steps up to a million rows. Every combination of complexity level and number of rows

was measured for 5 consecutive runs in isatools and 15 runs for isa4j (here results varied more) after a warm-up of writing 100 *Real World* complexity rows. Additionally, memory usage was measured for realistic complexity in 5 separate runs after CPU execution time measurements.

All evaluations were carried out on a Linux server with two Intel Xeon E5-2697 v2 CPUs running at 2.70 GHz, 256 GB DDR3 RAM running at 1600 MHz and CentOS 7.8.2003. isatools was evaluated under Python 3.7.3 [Clang 11.0.0 (clang-1100.0.33.16)] using isatools version 0.11 and memory-profiler version 0.57 for measuring RAM usage. isa4j was evaluated under AdoptOpenJDK 11.0.5. For both libraries, a memory consumption baseline was calculated after the warm-up runs and an additional Garbage Collector invocation. This baseline consumption was subtracted from all subsequent memory consumption values as we wanted to measure purely the memory consumed by the ISA-Tab content, not libraries and other periphery¹. The actual code generating the files and measuring time and memory usage for Python isatools² and isa4j³ can be found on the isa4j GitHub repository.

Results

Figure 2 shows the performance of both libraries at increasing file size for three different levels of complexity. isa4j consistently takes up less CPU execution time than isatools for all tested scenarios, reducing the time required for writing 1 million rows of Real World complexity from 8.6 hours to 43 seconds.

The emphasis on being useful especially in large-scale datasets is further amplified by isa4j's memory usage stability: While there is no notable increase for either library up to a volume of 25 rows, starting at about 250 rows, isatools memory consumption increases linearly with the number of rows being formatted, resulting in a maximum consumption of 15.8 GB for one million rows. isa4j memory consumption remains stable at about 0.5 MB independently of the number of rows written, demonstrating that the iterative technique of formatting and writing the rows had the desired effect.

Use Case: BRIDGE Web Portal

We have integrated isa4j into the BRIDGE portal, which is a visual analytics and data warehouse web application hosting data of 22621 genotyped and 9527 phenotyped germplasm samples of barley (*Hordeum vulgare* L.)¹². The underlying data was derived from the study of Milner *et al.*¹³. isa4j was integrated to allow the MIAPPE-compliant¹⁶ export of customized subsets of phenotypic data of germplasm samples together with the corresponding passport data¹⁴ in the ISA-Tab format. These

¹ Baseline memory consumption was approximately 100 MB for isatools and 11 MB for isa4j.

² https://github.com/IPK-BIT/isa4j/blob/master/src/test/resources/de/ipk_gatersleben/bit/bi/isa4j/performanceTests/isatools_performance_test.py

³ https://github.com/IPK-BIT/isa4j/blob/master/src/test/java/de/ipk_gatersleben/bit/bi/isa4j/performanceTests/PerformanceTester.java

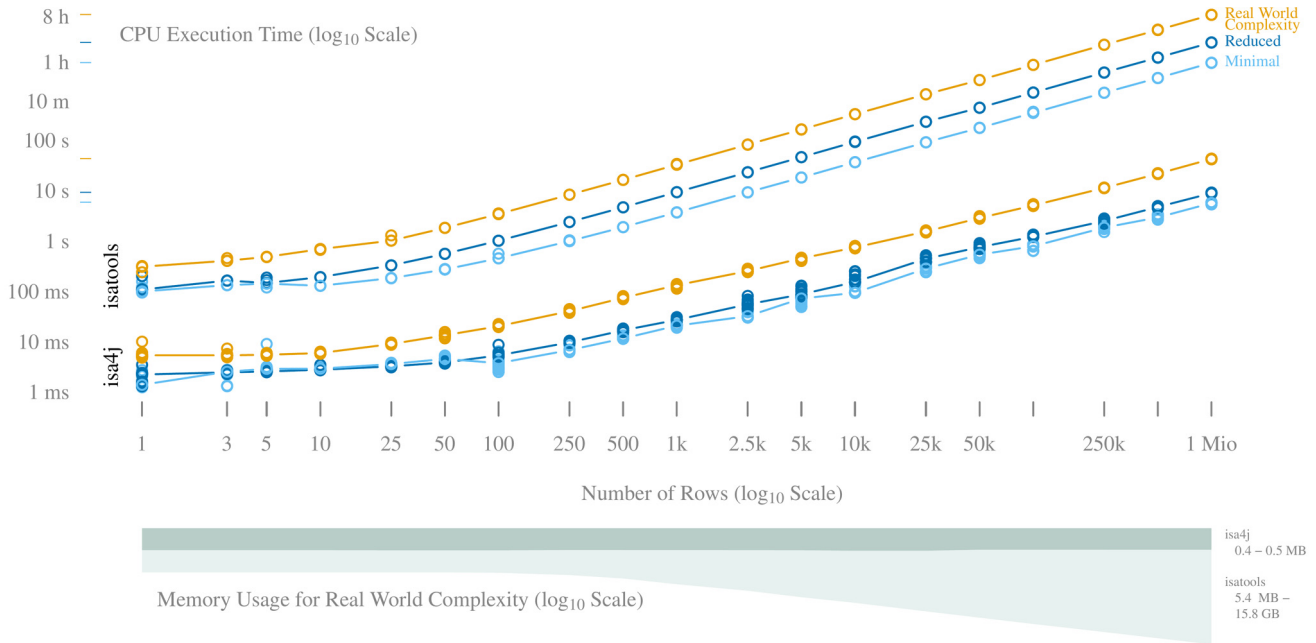


Figure 2. Performance comparison of isa4j and isatools. Top: CPU execution time at 3 different row complexity levels; Real World complexity was modelled after MIAPPE v1.1 compliant ISA-Tab metadata generated for a plant phenotyping experiment. Small colored lines on the left mark the highest point of each curve to help estimate the maximum value. Bottom: Memory consumption for isa4j and isatools along the same x-axis.

subsets can be derived from germplasm selections identified by the user during exploratory data analysis. In the ISA-Tab export dialog, the user can choose whether the associated plant images should be physically contained as files in the resulting ZIP file or whether they should only be linked as URLs to a version of the images available online. Due to the support of streaming in isa4j, the phenotypic data export module of BRIDGE is able to export large ZIP archives of several gigabytes with low main memory consumption of the web server. Another advantage over non-streaming approaches is that the download can start without delay and that no temporary files have to be created on the server. The process flow concept is shown in Figure 3.

Discussion

We have created a library for programmatically generating ISA-Tab metadata files in JVM-based environments and shown that it is considerably more performant and scalable than the existing Python based solution. It has been integrated into a large-scale data warehouse web software to validate practical feasibility and provide an example of how the library could help make ISA-Tab metadata available in time-sensitive applications.

CPU execution time appears to have a roughly linear relationship with the number of rows being written at $n > 250$ but this is only valid as long as isatools memory consumption does not surpass what the system can provide. Exceeding that, additional time for swapping from and to the hard disk will be required. There may also be further non-linear effects due to optimization steps, such as the compilation to native machine

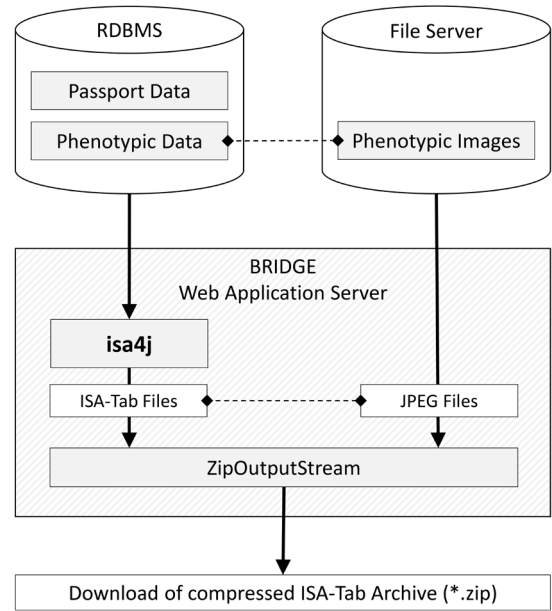


Figure 3. BRIDGE process flow. The BRIDGE web application uses isa4j to create ISA-Tab files from experimental data stored in a relational database management system (RDBMS). The data consists of passport data¹⁴ describing the basic characteristics of a germplasm sample (such as accession number and location of origin) on the one hand, and of phenotypic data, which systematically describe phenotypic characteristics of the individual germplasm samples, on the other. Phenotypic images that are stored as binary files on a separate file server are linked from the ISA-Tab files and are included in the final ISA-Tab ZIP archive.

code some JVMs perform for frequently used code parts. Lastly, exact CPU time requirements will naturally depend on the specific system in use but the overall relationships and proportions shown here should hold true for all situations.

Conclusions

The presented isa4j library provides a simple interface to create and export ISA-Tab metadata and can be seamlessly integrated into existing JVM-based pipelines, desktop tools or web applications. isa4j is less flexible than the Python-based isatools as it does not allow one to change the file structure after streaming has started, but the desired ISA-Tab configuration is often known beforehand, making this a peripheral limitation. In exchange, isa4j provides significantly better performance, especially for large datasets. We hope that this library will make the ISA framework available to an even wider audience and range of situations and help make published research data more interoperable and reusable for others. As a next step, we are going to begin developing a specialized isa4j extension for plant phenotyping experiments, isa4j-miappe, intended to make it

even easier for researchers in the field to ensure their metadata comply with the community standard. If you would like to contribute or develop an isa4j extension for your own community, please feel free to get in touch with us.

Data availability

Raw performance measurement data can be found at https://raw.githubusercontent.com/IPK-BIT/isa4j/master/docs/performance_data.csv (archived: Zenodo, IPK-BIT/isa4j: isa4j-1.0.4, <http://doi.org/10.5281/zenodo.4275168>¹⁵).

Software availability

Software available from: <https://mvnrepository.com/artifact/de.ipk-gatersleben/isa4j>

Source code available from: <https://github.com/IPK-BIT/isa4j>
Archived source code as at time of publication: <http://doi.org/10.5281/zenodo.4275168>¹⁵

License: MIT

References

- Barend M: **Invest 5% of research funds in ensuring data are reusable.** *Nature*. 2020; **578**(7796): 491–491.
[PubMed Abstract](#) | [Publisher Full Text](#)
- Wilkinson MD, Dumontier M, Aalbersberg IJJ, et al.: **The FAIR guiding principles for scientific data management and stewardship.** *Sci Data*. 2016; **3**(1): 160018.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
- Benson DA, Cavanaugh M, Clark K, et al.: **GenBank.** *Nucleic Acids Res*. 2018; **46**(D1): D41–D47.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
- The UniProt Consortium: **UniProt: a hub for protein information.** *Nucleic Acids Res*. 2014; **43**(Database issue): D204–D212.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
- Sansone S, Rocca-Serra P, Field D, et al.: **Toward interoperable bioscience data.** *Nat Genet*. 2012; **44**(2): 121–126.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
- Papoutsoglou EA, Faria D, Arend D, et al.: **Enabling reusability of plant phenomic datasets with MIAPPE 1.1.** *New Phytol*. 2020; **227**(1): 260–273.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
- González-Beltrán A, Neumann S, Maguire E, et al.: **The risa r/bioconductor package: integrative data analysis from experimental metadata and back again.** *BMC Bioinformatics*. 2014; **15** Suppl 1(Suppl 1): S11.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
- Haug K, Salek RM, Conesa P, et al.: **MetaboLights—an open-access general-purpose repository for metabolomics studies and associated meta-data.** *Nucleic Acids Res*. 2012; **41**(Database issue): D781–D786.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
- Rocca-Serra P, Brandizi M, Maguire E, et al.: **ISA software suite: supporting standards-compliant experimental annotation and enabling curation at the community level.** *Bioinformatics*. 2010; **26**(18): 2354–2356.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
- Maguire E, Gonzalez-Beltran A, Whetzel PL, et al.: **OntoMaton: a bioportal powered ontology widget for google spreadsheets.** *Bioinformatics*. 2010; **29**(4): 525–527.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
- Smith B, Ashburner M, Rosse C, et al.: **The OBO foundry: coordinated evolution of ontologies to support biomedical data integration.** *Nat Biotechnol*. 2007; **25**(11): 1251–1255.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
- König P, Beier S, Basterrechea M, et al.: **BRIDGE - a visual analytics web tool for barley genebank genomics.** *Front Plant Sci*. 2020; **11**: 701.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
- Milner SG, Jost M, Taketa S, et al.: **Genebank genomics highlights the diversity of a global barley collection.** *Nat Genet*. 2018; **51**(2): 319–326.
[PubMed Abstract](#) | [Publisher Full Text](#)
- Alercia A, Diulgheroff S, Mackay M: **FAO/Bioversity Multi-Crop Passport Descriptors V.2.1 [MCPD V.2.1]**. 2015.
[Reference Source](#)
- Psaroudakis D, Arend D: **IPK-BIT/isa4j: isa4j-1.0.4 (Version isa4j-1.0.4).** *Zenodo*. 2020.
<http://www.doi.org/10.5281/zenodo.4275168>

Open Peer Review

Current Peer Review Status:  

Version 1

Reviewer Report 08 March 2021

<https://doi.org/10.5256/f1000research.30037.r79165>

© 2021 Hoffmann 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.



Nils Hoffmann 

Center for Biotechnology (CeBiTec), Bielefeld University, Bielefeld, 33594, Germany

The authors describe a JAVA-based implementation of the Investigation-Study-Assay (ISA) framework for the structured description of biological experiments, their protocols and their results. They position their implementation as a complement to the existing Python-based implementation that uses a complete in-memory model of the ISA data structures before writing them to the actual output files. For large studies, e.g. for whole populations or large cohorts, this can mean that memory and CPU requirements are very demanding.

Thus, the authors implemented their JAVA library to write out lines as they arrive, requiring that the user fix their data format description before starting to write out to the final files. Therefore, their implementation's memory complexity scales constant with the number of rows to be written, as each row can be created ad-hoc and then written out to the target file. In order to underline this advantage over the Python-based library, the authors created different benchmarks, illustrating the memory and CPU time usage of each library for a collection of different study designs with increasing levels of complexity, highlighting the significant speed and memory advantage of their implementation.

Finally, the authors demonstrate the practical feasibility of their library through integration into the BRIDGE web portal where they employ isa4j to generate ISA-tab files on the fly for the studies stored in BRIDGE.

The support for ISA-tab in programming languages other than Python, especially with a focus on performance, is a timely and needed addition. For JAVA, the graphical client ISACreator was previously developed, but has not seen any significant updates throughout the last few years. Specifically, where neither the flexibility of the Python-based ISA tools nor a graphical user interface for a predefined ISA format are required, the isa4j library can be a valuable, performant, yet still validating tool to generate ISA-tab files in many different life-sciences domain, such as metabolomics, proteomics, genomics, etc. Thus, it addresses a current need and does this in a well designed and performant way.

Minor comments:

- The manuscript states, that the library is available from mvnrepository.com, while the GitHub page states that it is available from [Maven Central](https://maven.apache.org/), please update the manuscript accordingly.
- The online documentation to set up a custom isa4j project should also mentions that an slf4j logging implementation needs to be on the class path, e.g. for Gradle: implementation 'org.slf4j:slf4j-simple:1.7.30'
- The documentation at <https://ipk-bit.github.io/isa4j/getting-started/investigation-file.html> in section "Add them as investigation contacts" is syntactically not correct (missing semicolons ';'), also the method findByName does not exist in version 1.0.4 of the library (getByName does).
- I would recommend to set up a continuous integration system linked to the GitHub repository (GitHub actions, Travis, CircleCI, etc.) to make sure that the source code is buildable and automatically tested.

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

Yes

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?

Yes

Competing Interests: No competing interests were disclosed.

Reviewer Expertise: Bioinformatics, mass spectrometry of small molecules, data standardization.

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.

Reviewer Report 16 February 2021

<https://doi.org/10.5256/f1000research.30037.r77273>

© 2021 Izzo M. 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.



Massimiliano Izzo 

Oxford e-Research Centre, Department of Engineering Science, University of Oxford, Oxford, UK

The authors present isa4j, an optimised Java-based library to generate and serialise ISA-TAB metadata. I find particularly interesting that isa4j supports writing the ISA-metadata output on streams as well as files, as this can be very useful when building modern client-server applications. isa4j has an interesting approach of loading into memory only one row at the time, hence limiting memory consumption. The memory consumption comparison with isatools makes a good argument for using isa4j for certain large scale experiment.

I am curious to know whether isa4j-generated ISA-TABs comply with the [ISA-TAB validation rules](#), also with respect to the configuration files for specific assays (latest version can be found [here](#)). There are a few discrepancies with respect to the official ISA-TAB specifications: for instance, Processes cannot have names in "isa4j" and as a consequence "Assay Name" or synonymous columns are missing. Characteristic categories are treated as strings in isa4j, while are OntologyAnnotations in the ISA-API. I might have missed other, minor, discrepancies. I would suggest adding more equivalence tests with existing datasets to align more this library with isatools.

The developers don't seem to have put the tests into continuous integration; I think it would be worth doing so.

In any case, I think isa4j is a useful tool with strong performance, that will be very helpful to produce ISA-TAB metadata from a variety of large-scale experiments with noteworthy performances and low resources consumption.

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

Yes

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?

Yes

Competing Interests: No competing interests were disclosed.

Reviewer Expertise: Applied Computer Science, Software Engineering

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.

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