AIM: AN ABSTRACTION FOR IMPROVING MACHINE LEARNING PREDICTION

Victoria Stodden, Xiaomian Wu

University of Illinois at Urbana-Champaign
School of Information Sciences,
Department of Statistics
Champaign, IL 61820

Vanessa Sochat

Stanford University
Stanford Research Computing Center,
School of Medicine
Stanford, CA 94305

ABSTRACT

We introduce a structured and portable Abstraction for Improving Machine learning (AIM) to improve prediction outcomes and enable meaningful comparisons of ML pipelines. We implement AIM for a well-known acute leukemia classification problem using the Scientific Filesystem, enabling direct performance comparisons across a variety of classifiers. AIM provides three direct efficiency benefits: 1) the sources of performance differences between ML pipelines can identified at the algorithm implementation level as defined by the AIM, 2) improvements can be made to specific aspects of the pipeline and thus better understood, and 3) the reuse of these defined abstraction components across different pipelines is facilitated. When the AIM is defined at the outset of the prediction challenge, these benefits can come at minimal cost. We show these benefits by implementing AIM and the Scientific Filesystem on the well-known Golub AML/ALL cancer dataset.

Index Terms— reproducible research, programming abstraction, machine learning, Scientific Filesystem, containers, cyberinfrastructure.

1. INTRODUCTION

Machine Learning has become an indispensable tool for scientific advancement in a broad and increasing number of fields. This has brought a relatively new leaderboard style problem solving structure where a common dataset is analyzed by community members with the "winner" obtaining the lowest error rates on test data [1, 2, 3]. This approach raises an important question: Why does one learning approach perform better than another? In this paper provide a new way of assessing performance by proposing an Abstraction for Machine learning (AIM) that presents a structured delivery of the ML pipeline in leaderboard style competitions. AIM enables the direct evaluation and re-use of defined steps. We first motivate the need for AIM by examining a famous leukemia classification problem, then we implement structural innovations for prediction challenges via the AIM framework for this example that show the benefit of AIM. We utilize the Scientific Filesystem (SCIF) to create a Linux container that is structured according to the AIM, before any machine learning is attempted, that supports execution of components of the AIM.

2. BACKGROUND: THE LEADERBOARD APPROACH

The leaderboard approach has driven progress in natural language processing and biometrics research for the past two decades, and underlies the success of widely used speech recognition applications such as Apple's Siri and Amazon's Echo. In its most fundamental form the Machine Learning leaderboard approach possesses four features: a well-posed research question; a common dataset (with a holdout set not seen by researchers); the application of ML to build a predictive model; and a third party platform that tests the trained models on the holdout data and reports predictive accuracy. Some examples of platforms are Kaggle.com, DrivenData.org, OpenML.org, and CodaLab Competitions (https://competitions.codalab.org), and the Netflix [4] and the protein structure prediction [5] challenges. This setting makes clear that the variability of ML approaches and scientific outcomes can be very high. In one challenge, effect sizes varied from 0.89 to 2.93 in odds ratio units with 72% of the analyses using unique combinations of features [6]. As we will see in the next section, reconciling why one algorithm outperforms another requires the ability to compare algorithm implementations across the ML pipeline. The example we present in this article shows this to be a difficult but important task, improved by modularization and structure.

3. MOTIVATING AIM: CLASSIFYING CANCER

We are motivated by the the following query: List all of the classifiers applied to the famous leukemia dataset (AML/ALL), along with their error rates [7]. The goal is to evaluate classifier performance. We chose this query since this dataset is well-known and well-analyzed using many classifiers, and provides a pipeline that can be modularized into two components, providing a straightforward example of the Abstraction for Machine learning (AIM). This example also illustrates the need for AIM: without a modular structure for the computational pipeline, we show it is impossible to compare the performance of the classifiers used on this dataset.

In this problem gene expression data is used to discriminate between two cancers, acute myeloid leukemia (AML) and acute lymphoblastic leukemia (ALL). This dataset has 72 observations (25 AML and 47 ALL tumors) comprising 7192 genes. This is a challenging high-dimensional supervised learning problem with a small number of observations relative to a much greater number of variables, in addition to classical measurement error and biological noise. However, most genetic features will not be related to the cancer type which reduces the dimensionality of the problem [8]. We set the notation as follows. Let $X = (x_{ij})$ be the the dataset of genetic predictor variables, where x_{ij} is the expression of gene j in sample i. $\mathbf{x}_i = (x_{i1}, \ldots, x_{ip})$ is the gene expression profile for sample i. y_i is the response or class label, for 2 classes. Let $\mathcal{X}$ be the space of all gene expression profiles. Let $\mathcal{L} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_{n_\mathcal{L}}, y_{n_\mathcal{L}})\}$ be the learning set, $\mathcal{T} = \{(\mathbf{x}_{n_{\mathcal{L}+1}}), \ldots, (\mathbf{x}_n)\}$ be the test set, and $\mathcal{C} = \mathcal{C}(\mathbf{x}, \mathcal{L})$ be our classifier. Our classification problem can be stated as follows: Given a learning set $\mathcal{L} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_{n_\mathcal{L}}, y_{n_\mathcal{L}})\}$ where the $\mathbf{x}_i$'s are independent p-dimensional gene expression samples and the y_i's the class labels; and given a test set $\mathcal{T} = \{(\mathbf{x}_{n_{\mathcal{L}+1}}), \ldots, (\mathbf{x}_n)\}$, find a classification function $\mathcal{C} = \mathcal{C}(\cdot, \mathcal{L})$ that maximizes classification accuracy on $\mathcal{T}$.

This first step is answering the query is to discover which classifiers have been applied to the AML/ALL dataset. A literature search produced 30 publications that did so. We found that a direct comparison of reported classifier performance was impossible: different preprocessing and feature selection steps were taken in each of the papers, meaning the baselines for classifier comparison changed from paper to paper. We chose five articles we deemed computationally reproducible (these are papers 1 [9], 3 [10], 6 [11], 9 [12], and 29 [13]). We note that these are excellent articles and we make no criticism of of the work in isolation. (See `https://github.com/AIM-Project/AIM-Manuscript` for a summary of all classifiers used in the 30 publications and flow charts of the computational pipelines for the five articles we replicated).

The impossibility of direct classifier comparisons for this datasets highlights shortcomings in the structure of publications. Some work is emerging to establish ML pipelines for some codes, for example scikit-learn pipelines and TensorFlow's visualization module, and in this article we present a general concept of abstraction in the ML context and demonstrate its utility in permitting comparisons and improved prediction outcomes regardless of the software or computer system used.

In the cancer classification example we held the preprocessing and feature selection pipeline component (PPFS) constant then trained each classifier on each PPFS component in turn. This created a meta-analysis validation table [14] presented in Table 1. The columns give the preprocessing and feature selection pipeline used (the 6th paper uses two separate feature selection methods), and the rows indicate the classifier applied. Table 1 shows that feature selection is a key driver of performance in this high-dimensional data setting while no one classifier emerges as uniformly superior. The best average performance of a classifier across PPFS methods in this sample of five articles is logistic discrimination in [11] at 96% (note that four other methods score 95% classification accuracy). The highest average accuracy score for preprocessing / feature selection across classification algorithms is also from [11], using Partial Least Squares for dimensionality reduction with 97%. These insights are not readily apparent until the models are compared as in Table 1. Our classification problem can now be revised: Find a classification function $\mathcal{C} = \mathcal{C}(\cdot, \tilde{L})$ that maximizes classification accuracy on $\tilde{\mathcal{T}}$, where $\mathcal{F}(Z) = \tilde{Z}$ is a function that carries out preprocessing and feature selection steps on input data Z.

	Preprocessing / Feature Selection						
Paper/Classifier	1	3	6a	6b	9	29	Ave
1WeightedVote	.91	.94	.97	.97	.89	.74	.90
3NN	.97	.94	.91	.94	.97	.97	.95
3SVM Linear	.97	.97	.94	.97	.97	.77	.93
3SVM Quad	.97	.88	.97	.97	.97	.91	.95
3Adaboost	.91	.91	.97	.97	.91	.91	.93
6Logit	.97	.97	.97	.97	.97	.88	.96
6QDA	.94	.91	.94	.97	.97	.85	.93
9NN	.97	.91	.85	.97	.94	.94	.93
9Decision Tree	.91	.91	.97	.97	.91	.77	.90
9Bagging	.94	.91	.97	.97	.92	.77	.91
9Bagging (CPD)	.74	.85	.82	.91	.77	.68	.79
9FLDA	.88	.88	.97	.97	.88	.88	.91
9DLDA	.97	.94	.97	.97	.97	.88	.95
9DQDA	.97	.94	.97	.97	.97	.88	.95
29BayesNetwork	.74	.88	.97	.97	.83	.62	.83
Average	.92	.92	.95	.97	.92	.83	

Table 1. Classification accuracy for the AML/ALL dataset, when preprocessing and feature selection steps are held constant for each classifier, allowing for the direct comparison of the classifiers presented in the five publications studied.

4. AIM: A GENERAL ABSTRACTION LAYER FOR THE LEADERBOARD APPROACH

The notion of an abstraction representing a more complex underlying reality is a fundamental concept in computer science [15, 16, 17]. In this work we apply the concept of an abstraction explicitly to machine learning code, to separate parts of the computational pipeline that are distinct steps for a particular prediction problem. For example, a raw dataset may have had some feature selection techniques as we saw, such as principle components analysis (PCA) or multidimensional scaling (MDS), applied prior to the training of a machine learning algorithm. Ensemble methods may be applied to combine various machine learning techniques, for example. We propose that such steps are defined prior to an ML leaderboard competition to aid in the reconciliation of prediction differences by different computational pipelines and to enable the efficient re-use of specific modules, such as feature selection.

Building on work defining Experiment Definition Systems [18, 19, 20, 21], we propose the use of a formal abstraction layer, called AIM (Abstraction for Improving Machine learning), that pre-specifies the modular steps in the ML pipeline. Experiment Definitions Systems propose that the computational environment with which the user interacts is tailored for scientific applications.

In the ALL/AML example the abstraction layer specifies data preprocessing and feature selection, and classifier model building. A key point is that each component of the AIM layer is implemented as standalone code (for example a standalone function or Jupyter Notebook) taking outputs from other components as inputs like a highly specialized set of functions. This allows three direct benefits for ML leaderboard challenges: 1) the sources of differences between ML pipelines can be traced at the implementation level, 2) improvements can be made to specific aspects of the workflow, for example feature selection or ML algorithm, 3) the reuse of components across pipelines is facilitated. This structured abstraction layer also permits a localized investigation into sources of bias and uncertainty in prediction and can provide checks on whether model implementation is correct, for example whether pre-validation was appropriately used [22]. The routine use of the AIM in ML leaderboard competitions would permit a greater understanding of the underlying scientific phenomena and the behavior of ML algorithms, as suggested by the AML/ALL example. We found generating Table 1 to be time consuming for these five articles (using more than 200 student hours) and impossible for the other 25 articles due to missing implementation details. In ML leaderboard challenges, the implementation is made available to the hosting platform by design, giving a unique opportunity to automate the controlled comparison as in Table 1 through the use of a well-defined abstraction layer.

5. DEFINING THE AIM FOR THE AML/ALL CANCER DATASET

We have created an example AIM for Paper1 [9] with a component for the preprocessing and feature selection steps, and a component for the classifier building steps ($\mathcal{F}_1$ and $\mathcal{F}_2$). We created a Jupyter notebook for each, available at `https://github.com/AIM-Project/AIM-Manuscript`.

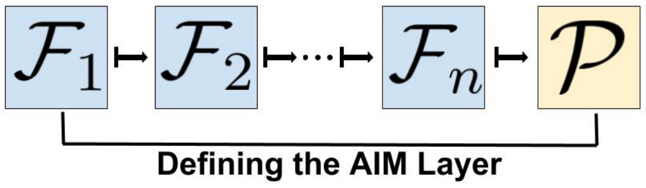


Fig. 1. A cartoon AIM layer showing a segmentation of ML pipeline into discrete components $\mathcal{F}_1, \ldots, \mathcal{F}_n$ that carry out n data steps to be input into a prediction model $\mathcal{P}$. Steps may include searches over parameter spaces, for example. The AIM layer can be adapted to various ML pipelines facilitating comparisons and re-use of components.

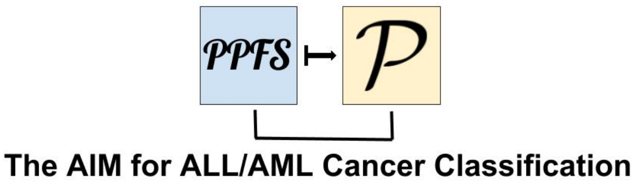


Fig. 2. We show a graphical representation of the simple AIM we defined for the Golub cancer papers we reproduced in this article. It shows the segmentation of the workflow into two discrete components: Preprocessing/Feature Extraction (PPFS) and Classifier (P), matching the layout in Table 1.

We can now describe a general AIM layer in the leaderboard setting. We now define our ML challenge as follows: Find the function $\mathcal{P} = \mathcal{P}(\cdot, \tilde{L})$ that maximizes prediction accuracy on test data $\tilde{\mathcal{T}}$ where $\tilde{Z}$ is the output of a sequence of functions $\mathcal{F}_1, \ldots, \mathcal{F}_n$ that carry out n steps on input data Z. By design, the $\mathcal{F}_i$ are modular, in that a change to $\mathcal{F}_i$ will not change $\mathcal{F}_j, \forall j \neq i$. This independence underpins the value of the AIM layer: prediction accuracy will be maximized if each component $\mathcal{F}_i$ in the abstraction layer is optimized for a particular $(X, \mathcal{P})$ pair. The AIM layer provides a structured way to track and isolate errors and evaluate hardware utilization, local algorithm performance (e.g. for an AIM component $\mathcal{F}_i$ or $\mathcal{P}$), and resource use. It also permits users to visualize defined steps or components in the ML pipeline, from preprocessing to prediction.

6. IMPLEMENTING AIM USING THE SCIENTIFIC FILESYSTEM

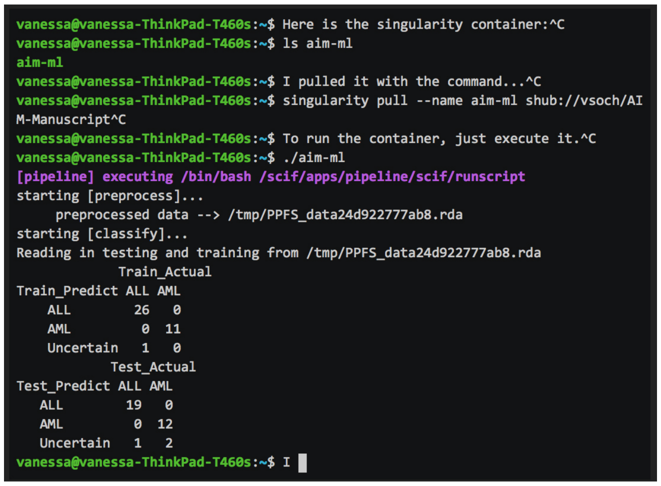
As an abstraction, the AIM layer itself is defined independently of a particular computational implementation or system. This is what enables AIM to facilitate the comparison of pipeline components (the the entire pipeline) that have been implemented in different ways or on different systems. As Linux containers such as Singularity and Docker [23, 24] have become more popular, a natural step is to embody the AIM abstraction using containerization to enable portability and executability of the implementation of the machine learning pipeline, as well as pipeline comparison. The AIM abstraction, once defined, can be implemented as a Scientific Filesystem in a container that then will also carry metadata, parameters and input values, execution instructions, software versions and libraries, and other information, as well as the scripts and code that implement the machine learning pipeline [25]. SCIF is a specification for modular organization of content, which can be constructed to match the AIM.

We defined a Scientific Filesystem in a container for the original Golub paper (paper1) in our cancer data example, using the AIM defined in Figure 2. In this case subfolders are created within the container structure that correspond to the AIM, with one subfolder for each component of the AIM. In this Scientific Filesystem example there are two subfolders, one for the preprocessing and feature selection component PPFS (called preprocess/) and one for training the classifier (called classify/). There is a third subfolder called pipeline for running the ML pipeline for paper1. We then added a Jupyter Notebook for each AIM component in each subdirectory. Each component of the AIM can be run by itself from the container implementation of the Scientific Filesystem. See `https://github.com/vsoch/AIM-Manuscript/tree/master/ReproducingMLpipelines/Pipelines/PipelineExample` to access the container as well as a tutorial showing how to run the ML pipeline or the components defined by the AIM.

7. CONCLUSION

The Abstraction for Improving Machine learning (AIM) layer proposes a way to improve the performance of leaderboard style ML challenges by structuring submissions to leverage additional relevant information. The abstraction is customizable for each challenge to support creativity and varied approaches, but is fixed within a challenge to enables the comparability and modularized re-use of components of ML prediction pipelines. We propose combining the AIM with the same test/train data division for all competition participants where they only see the training data, as is typical for ML leaderboard challenges. We show that without modularizing the computational pipeline into components as done by the AIM, it is impossible to directly compare the published performance of the classifiers used for the AML/ALL dataset.

Future work could test the prediction improvement of the AIM in different settings by deploying container images that are preloaded with the Scientific Filesystem corresponding to a particular AIM in an ML Prediction Challenge. This could be done in a controlled way by designing an experiment as follows. For a dataset, define one Challenge using AIM/SCIF and one that does not. This could be done in a classroom setting where students are randomly divided into treatment and control groups. We could see which group obtains the "winning" pipeline and whether this depends on information sharing between pipelines.



```
vanessa@vanessa-ThinkPad-T460s:~$ Here is the singularity container:^C
vanessa@vanessa-ThinkPad-T460s:~$ ls aim-ml
aim-ml
vanessa@vanessa-ThinkPad-T460s:~$ I pulled it with the command...^C
vanessa@vanessa-ThinkPad-T460s:~$ singularity pull --name aim-ml shub://vsoch/AI
M-Manuscript^C
vanessa@vanessa-ThinkPad-T460s:~$ To run the container, just execute it.^C
vanessa@vanessa-ThinkPad-T460s:~$ ./aim-ml
[pipeline] executing /bin/bash /scif/apps/pipeline/scif/runscript
starting [preprocess]...
     preprocessed data --> /tmp/PPFS_data24d922777ab8.rda
starting [classify]...
Reading in testing and training from /tmp/PPFS_data24d922777ab8.rda
             Train_Actual
Train_Predict ALL AML
    ALL        26   0
    AML         0  11
    Uncertain   1   0
            Test_Actual
Test_Predict ALL AML
   ALL        19   0
   AML         0  12
   Uncertain   1   2
vanessa@vanessa-ThinkPad-T460s:~$ I
```

Fig. 3. We show a screen shot of the container executing the Golub pipeline (paper1) in a container using the Scientific Filesystem with preprocess/ and classify/ subdirectories corresponding to the AIM defined in Figure 2. The code in each subdirectory can execute a component of the AIM independently as well. Available at `https://github.com/vsoch/AIM-Manuscript/tree/master/ReproducingMLpipelines/PipelineExample`.

8. REFERENCES

[1] Isabelle Guyon, Amir Reza Saffari Azar Alamdari, Gideon Dror, and Joachim M. Buhmann, "Performance prediction challenge," in *Proceedings of the International Joint Conference on Neural Networks, IJCNN 2006, part of the IEEE World Congress on Computational Intelligence, WCCI 2006, Vancouver, BC, Canada, 16-21 July 2006*, 2006, pp. 1649–1656.

[2] Isabelle Guyon, Constantin Aliferis, Greg Cooper, and Andre Elisseeff, "Design and analysis of the causation and prediction challenge," in *Proceedings of the Workshop on the Causation and Prediction Challenge at WCCI 2008*, Hong Kong, 03–04 Jun 2008, vol. 3 of *Proceedings of Machine Learning Research*, pp. 1–33, PMLR.

[3] Christopher K Wikle, Noel Cressie, Andrew Zammit-Mangion, and Clint Shumack, "A common task framework (ctf) for objective comparison of spatial prediction methodologies," *Statistics Views*, 2017.

[4] "Netflix prize.," 2009.

[5] "Critical assessment of protein structure prediction.," 2016.

[6] Raphael Silberzahn et al., "Many analysts, one dataset: Making transparent how variations in analytical choices affect results," *Advances in Methods and Practices in Psychological Science*, vol. 1, no. 1, 2018.

[7] Matan Gavish, David L. Donoho, and Amos Onn, "Dream applications of verifiable computational results," *ACM Crossroads*, vol. 19, no. 3, pp. 58–61, 2013.

[8] Ying Lu and Jiawei Han, "Cancer classification using gene expression data," *Inf. Syst.*, vol. 28, no. 4, pp. 243–268, June 2003.

[9] T. R. Golub, D. K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. P. Mesirov, H. Coller, M. L. Loh, J. R. Downing, M. A. Caligiuri, C. D. Bloomfield, and E. S. Lander, "Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring," *Science*, vol. 286, no. 5439, pp. 531–537, 1999.

[10] Amir Ben-Dor, Laurakay Bruhn, Nir Friedman, Iftach Nachman, Michèl Schummer, and Zohar Yakhini, "Tissue classification with gene expression profiles," in *Proceedings of the Fourth Annual International Conference on Computational Molecular Biology*, New York, NY, USA, 2000, RECOMB '00, pp. 54–64, ACM.

[11] Danh V. Nguyen and David M. Rocke, "Classification of acute leukemia based on dna microarray gene expressions using partial least squares," in *Methods of Microarray Data Analysis: Papers from CAMDA '00*, Simon M. Lin and Kimberly F. Johnson, Eds., pp. 109–124. Springer US, Boston, MA, 2002.

[12] Sandrine Dudoit, Jane Fridlyand, and Terence P. Speed, "Comparison of discrimination methods for the classification of tumors using gene expression data," *Journal of the American Statistical Association*, vol. 97, no. 457, pp. 77–87, 2002.

[13] Abdel Nasser Zaied, Mona G. Habishy, and Mohamed A. Saleh, "Acute leukemia classification using bayesian networks," *International Journal of Emerging Trends in Computing and Information Sciences*, vol. 3, pp. 1419–1426, 10 2012.

[14] Levi Waldron et al., "Comparative meta-analysis of prognostic gene signatures for late-stage ovarian cancer," *JNCI: Journal of the National Cancer Institute*, vol. 106, no. 5, pp. dju049, 2014.

[15] Luca Mottola and Gian Pietro Picco, "Logical neighborhoods: A programming abstraction for wireless sensor networks," in *Distributed Computing in Sensor Systems*, Phillip B. Gibbons, Tarek Abdelzaher, James Aspnes, and Ramesh Rao, Eds., Berlin, Heidelberg, 2006, pp. 150–168, Springer Berlin Heidelberg.

[16] P. A. Vicaire, E. Hoque, Z. Xie, and J. A. Stankovic, "Bundle: A group-based programming abstraction for cyber-physical systems," *IEEE Transactions on Industrial Informatics*, vol. 8, no. 2, pp. 379–392, May 2012.

[17] Allen Leung, Nicolas Vasilache, Benoît Meister, Muthu Baskaran, David Wohlford, Cédric Bastoul, and Richard Lethin, "A mapping path for multi-gpgpu accelerated computers from a portable high level programming abstraction," in *Proceedings of the 3rd Workshop on General-Purpose Computation on Graphics Processing Units*, New York, NY, USA, 2010, GPGPU-3, pp. 51–61, ACM.

[18] Hatef Monajemi, David L. Donoho, and Victoria Stodden, "Making massive computational experiments painless," in *2016 IEEE International Conference on Big Data, BigData 2016, Washington DC, USA, December 5-8, 2016*, 2016, pp. 2368–2373.

[19] Ivo Jimenez, Michael Sevilla, Noah Watkins, Carlos Maltzahn, Jay Lofstead, Kathryn Mohror, Andrea Arpaci-Dusseau, and Remzi Arpaci-Dusseau, "Standing on the shoulders of giants by managing scientific experiments like software," in *USENIX*, 2016, vol. 41.

[20] Thomas Quinn, Daniel Tylee, and Stephen Glatt, "exprso: an r-package for the rapid implementation of machine learning algorithms," *F1000Research*, vol. 5, no. 2588, 2017.

[21] Ulrich Mansmann, Markus Ruschhaupt, and Wolfgang Huber, *Reproducible Statistical Analysis in Microarray Profiling Studies*, pp. 939–948, Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

[22] Robert J. Tibshirani and Brad Efron, "Pre-validation and inference in microarrays," *Statistical Applications in Genetics and Molecular Biology*, , no. 1, 2002.

[23] Dirk Merkel, "Docker: Lightweight linux containers for consistent development and deployment," *Linux J.*, vol. 2014, no. 239, Mar. 2014.

[24] Abdelrahman Hosny, Paola Vera-Licona, Reinhard Laubenbacher, and Thibauld Favre, "Algorun: a docker-based packaging system for platform-agnostic implemented algorithms," *Bioinformatics*, vol. 32, no. 15, pp. 2396–2398, 2016.

[25] Vanessa Sochat, "The scientific filesystem (scif)," *GigaScience*, p. 100420, 2018.