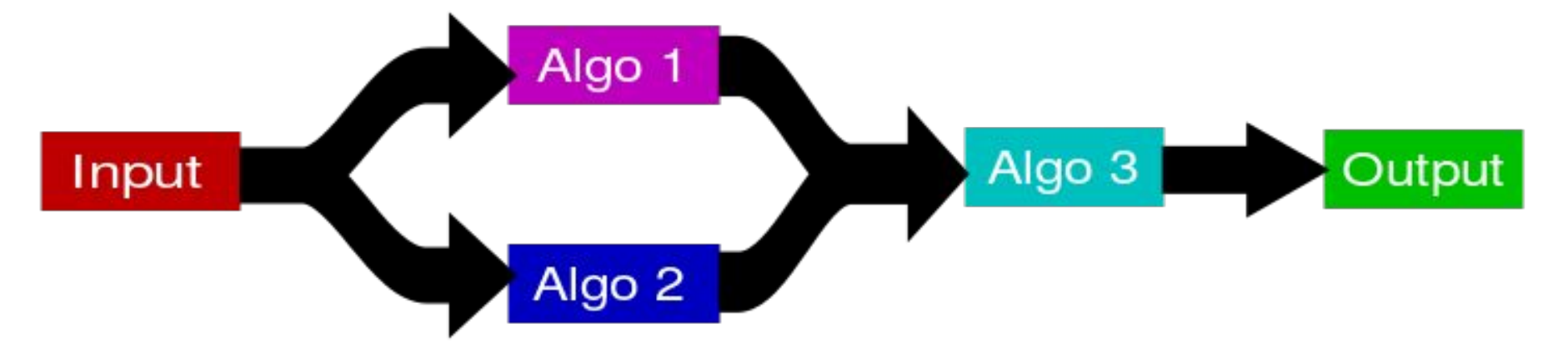
Building Reproducible and Shareable Pipelines in Computational Biology

Vishal Shah[1], Paola Vera-Licona[2]
[1]Avon High School, [2]Center for Quantitative Medicine, UConn Health

Introduction

Motivation:

A computational workflow or pipeline is a description of a series of computational steps connected to each other. Each step accepts one or more input(s) and produces one or more output(s). Computational biology and bioinformatics analyses increasingly rely on pipelines to process diverse high-throughput data and model complex systems.



Example of a pipeline

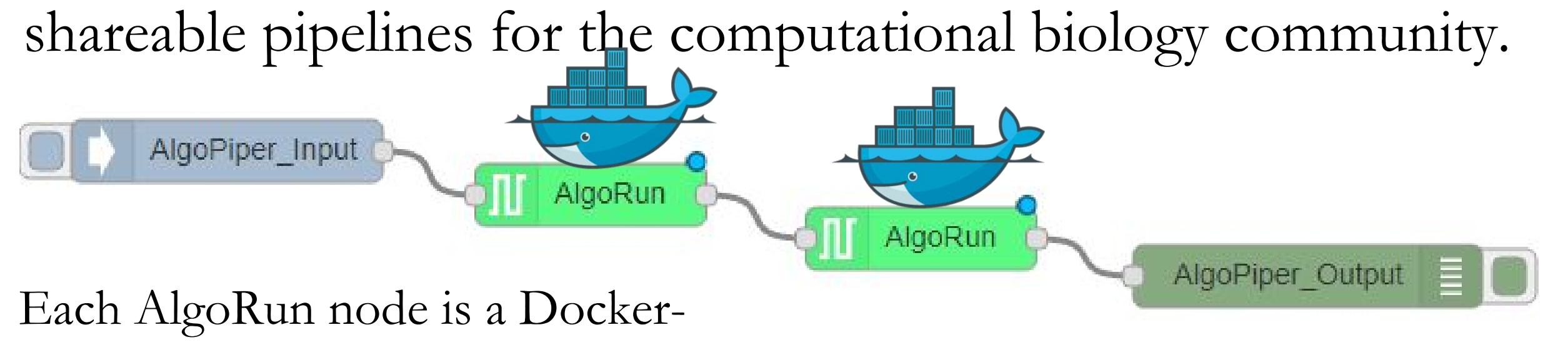
Despite the usefulness of pipelines, there are some complications when it comes to building pipelines. Reproducing pipelines is especially difficult because of their complexity. In fact, a study was done in which it was attempted to reproduce the results of a published computational analysis, and it took 280 hours to do so [1]. Sharing pipelines is also extremely difficult for similar reasons. A possible way to share pipelines is through Github [2], but it is not guaranteed to be executable.

As a result, we want an easy systematic way to share computational pipelines and algorithms. We want experts to be able to share pipelines, and want non-experts to be able to run them.

With the development of containerized technology, several tools already exist that allow for pipelining of algorithms and container orchestration such as Nextflow [3] and Bioboxes [4]. However, Nextflow works entirely in the command line which makes it difficult to use for users without as much technical expertise, and Bioboxes has a great deal of specific protocol, making the learning curve much greater.

Approach:

With the use of the recently proposed, Docker [5] container-based algorithm packaging solution AlgoRun [6], and with newly developed, Node-RED [7] based, AlgoPiper [8], we will show how to easily create reproducible and shareable pipelines for the computational biology community.



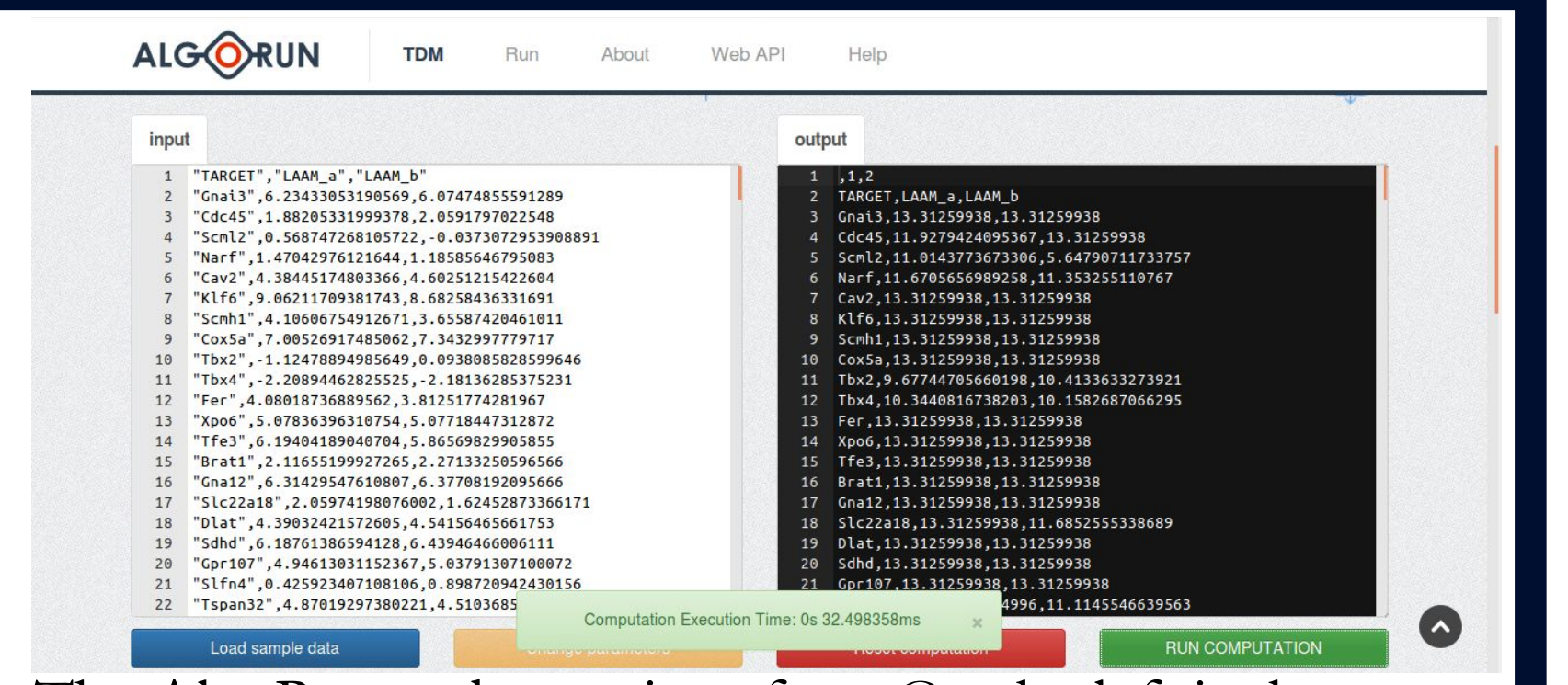
Each AlgoRun node is a Docker-based packaged algorithm to allow for modularization of pipelines

Methods

ALGORUN

Docker [5] is an emerging tool that allows users to create software containers. Docker containers appear to be a viable alternative to virtual machines. Because they do not have an entire operating system and use the host OS's kernel, containers are significantly faster and smaller than virtual machines [5]. Algorun utilizes Docker to create containers with a web interface and other features. AlgoRun's features include:

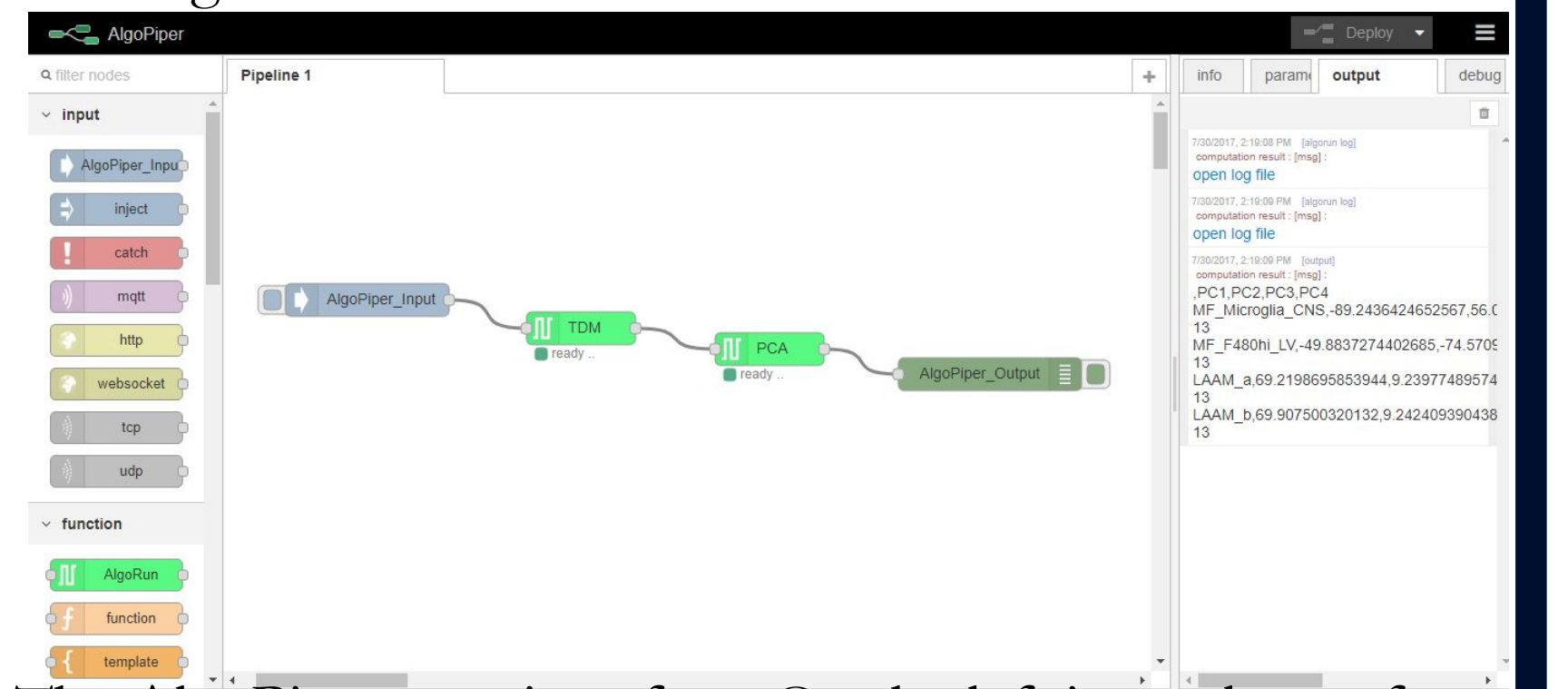
- Remote execution ability
- Ease-of-use using graphical web interface
- Portability and compatibility
- Programmability and reusability



The AlgoRun web user interface. On the left is the input textbox and on the right is the output textbox. The run computation button on the bottom right runs the algorithm

ALGOPIPER

Node-RED is a graphical tool, originally developed by IBM, for wiring together hardware devices, API's, and online solutions. Node-RED incorporates a web interface that allows for the pipelining of several nodes using drag-and-drop technology [7]. AlgoPiper utilizes Node-RED's capabilities, and implements the ability to pipeline AlgoRun package nodes in the user interface [8].



The AlgoPiper user interface. On the left is a palette of actionable nodes that includes AlgoRun nodes. These nodes have a drop-down menu from which we can select an AlgoRun packaged algorithm. On the right is the output and in the middle is the pipeline.

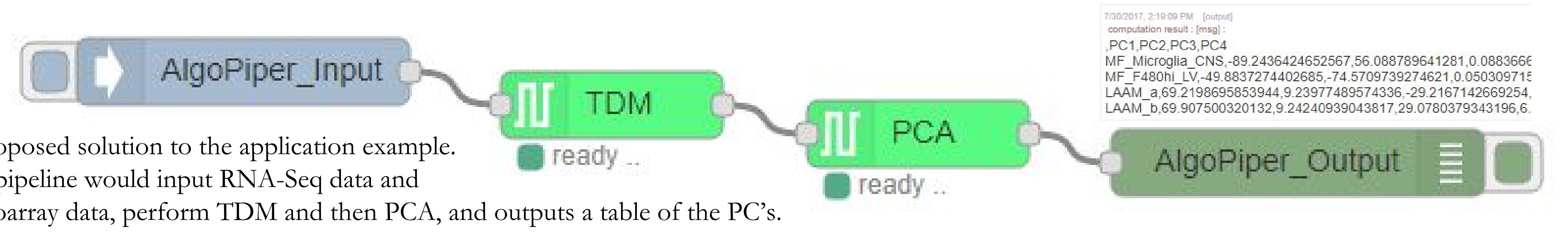
Application Example

This performs cross-platform comparison of gene-expression patterns in tissue-resident macrophages. Macrophages are a type of white blood cell that "eat" menacing cells in an organism's body. Macrophages that reside in different tissues can have different characteristics, making them different types of macrophages.

With the use of RNA-Seq and microarray technologies, scientists have studied gene-expression patterns in many cells. Currently, investigators are increasingly measuring gene expression using RNA-Seq due to its advantages (it does not require prior knowledge of the gene sequence, it is able to detect single nucleotide variations, etc.) [9,10]. However, there is still a plethora of microarray data that is available for use. A new tool, Training Distribution Matching (TDM), was proposed to normalize RNA-Seq data to allow models trained on microarray data to be used on RNA-Seq [11]. This enables RNA-Seq data to be comparable with microarray data.

Principal Component Analysis (PCA) is an algorithm that reduces the dimensionality of a dataset while retaining the variation of the dataset. It identifies directions, called Principal Components (PC), where variation is maximal. Using a few components allows the data to be represented neatly on a graph, making it easier to gauge similarities and differences in the samples [12].

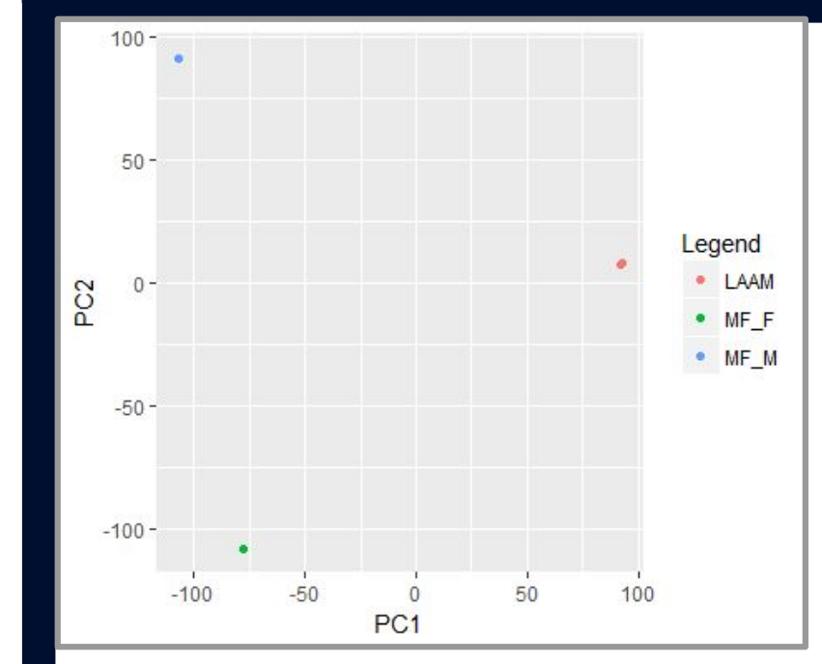
In this case study, we want to observe whether gene expression patterns in a tissue resident, *in vivo* macrophage population that resides in the lungs, named Lung Airways Associated Macrophages (LAAM), which are newly discovered by our collaborator, Dr. Khanna, are different from the gene-expression patterns from other tissue-resident macrophages such as microglia in the central nervous system (CNS) and F4/80 liver macrophages. To that end, we utilized RNA-seq data from LAAM populations from Khanna's lab and microarray data from the Immgen database for the microglia and F4/80 liver macrophages [13,14].



```
7/30/2017, 2:19:09 PM   [output]
computation result : [msg] :
,PC1,PC2,PC3,PC4
MF_Microglia_CNS,-89.2436424652567,56.088789641281,0.0883666
MF_F480hi_LV,-49.8837274402685,-74.5709739274621,0.05030971
LAAM_a,69.2198695853944,9.23977489574336,-29.216714266925
LAAM_b,69.907500320132,9.24240939043817,29.0780379343196,6.
```

A proposed solution to the application example. The pipeline would input RNA-Seq data and microarray data, perform TDM and then PCA, and outputs a table of the PC's.

Results



A graph of the PC's. The MF_M points are microglia, and the MF_F points are F4/80 liver macrophages. This PCA plot was a result of the AlgoRun-packaged PCA algorithm

We successfully packaged TDM and PCA in AlgoRun, and then we created an AlgoPiper pipeline to connect them together. The inputs of the pipeline are the RNA-seq and microarray datasets. The output passed to PCA from TDM is the re-normalized RNA-seq data. With the re-normalized RNA-seq and the original microarray datasets, the PCA provides the final output of the pipeline as a table of Principal Components.

After processing the data in our pipeline, we observe that the two LAAM samples indeed cluster together, while the microglia and F4/80 liver macrophages cluster far apart, indicating a large transcriptional difference between these three macrophage populations.

Discussion

As seen, AlgoRun and AlgoPiper are both useful tools to package and pipeline shareable algorithms in an easy way. However, AlgoPiper could benefit from some improvements. For instance, the implementation of multiple inputs and outputs as well as the ability to output graphics would be beneficial improvements to AlgoPiper.

In the end, however, AlgoRun and AlgoPiper are useful tools that allow for better reproducibility and shareability of computational algorithms and pipelines.

Conclusion

The issue of computational reproducibility is far too simple to be an issue of this magnitude, considering the ability for computers to automate processes. The reproduction of algorithms and pipelines can be completely automated. It costs researchers valuable time and energy to recreate these pipelines. With AlgoRun and AlgoPiper, this issue will be solved by using modular containers to package algorithms with all of their dependencies.

References

[1] D Garjio et al. (2013) PLoS one 8(11):e80278
[2] http://www.github.com
[3] P. Di Tommaso et al. (2017) Nature Biotechnology 11;35(4):316-319
[4] P. Belmann et al. (2015) Gigascience 15;4:47
[5] http://www.docker.com
[6] A. Hosny et al. (2016) BioInfomatics 1;32(15):2396-8
[7] http://www.nodered.org
[8] http://www.algopiper.org
[9] Z. Wang et al. (2009) Nature reviews. Genetics. 10(1):57-63
[10] ZK. Atak et al. (2013) PLoS Genetics 9(12):e1003997
[11] JA. Thompson et al. (2016) PeerJ 4:e1621
[12] http://www.nature.com
[13] B. Cicek et al. Paper in preparation
[14] TS. Heng, et al. Immunological Genome Project Consortium. Nat Immunol. 2008 10:1091

Acknowledgements

Funded by Achieve: Avon High School Internship Program and the Center for Quantitative Medicine at UConn Health