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 pipeline is a chain of programs in which the output of a program is the input of the next. They allow complex programs to be broken down into simpler segments and allow for advanced algorithms to be created. However, sharing and reproducing pipelines in computational biology is extremely time-consuming. A study was performed which studied the amount of time it takes to reproduce the results of an experiment, and it found that it takes 280 hours to do so. This is a result of the complexity of computational workflows. With dozens of dependencies, they are very complex and difficult to set up, especially when running them on varying execution environments. Software containers, such as Docker, are an emerging tool that provides a uniform execution environment for running a program. _Containers are faster and smaller than virtual machines, the other solution to this issue_. Several tools already exist that allow for pipelining of algorithms and container orchestration such as Nextflow () and Bioboxes (). 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.

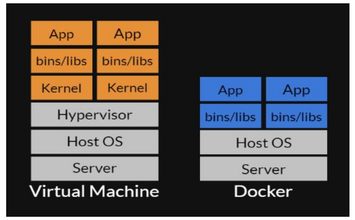
Approach:

To solve this issue, we created AlgoRun () and AlgoPiper (). These tools allow users to package and pipeline algorithms with a user interface and very simple protocol. With the use of these two newly proposed tools, shareability and reproducibility of pipelines is greatly increased. __ In a case study, a pipeline is created for cross-platform gene expression analysis, further showing the applicability and simplicity of AlgoRun and AlgoPiper.

Methods

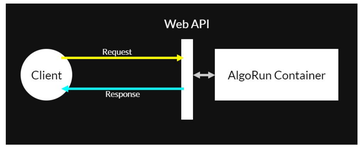
Docker Containers

Docker 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.



AlgoRun & AlgoPiper

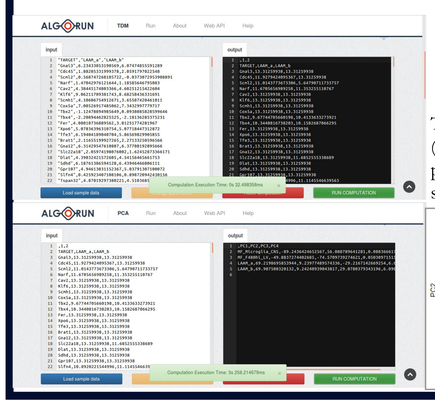
AlgoRun and AlgoPiper improve upon Docker's capabilities, and allow for the creation of pipelines. AlgoRun adds more modularity and features to Docker, while AlgoPiper allows for pipelining of AlgoRun packages. Both AlgoRun and AlgoPiper have web user interfaces, which make it easier for less experienced users to use and allows for remote execution of algorithms. In addition, since they are Docker-based, they also gain the main benefits of Docker, including portability and compatibility of softwares. Both softwares utilize the RESTful API, which allows for modularity and implementation of algorithms in larger softwares.



Results

This pipeline performs cross-platform gene expression analysis where RNA-seq data is first converted into microarray data (using TDM). Then, principal component analysis is performed on this new data to produce a data table of all of the principal components. Unfortunately, AlgoPiper cannot handle graphics, so the pipeline only outputs a data table.

The sample dataset used here contains microarray data from the microglia and f4/80 macrophages, which abide in the central nervous system and the liver, respectively () (). The RNA-seq data contains gene expression from a newly discovered type of macrophage, the Lung Airways Associated Macrophage (LAAM), which live in the lungs. The data performed by PCA indicates that LAAM macrophages are indeed different from microglia and f4/80 macrophages.



TDM (top) and PCA (bottom) AlgoRun packages running separately.

pipeline

Discussion

As seen, it is extremely easy to package and pipeline shareable algorithms. Many more applications to this pipeline exist beyond macrophages, and even more applications of AlgoRun and AlgoPiper exist.

AlgoRun and AlgoPiper both support portability of algorithms which removes the issue of reproducibility, and makes it easier to share algorithms and pipelines.

Being able to pipeline AlgoRun algorithms (using AlgoPiper) allows for easy execution of pipelines, where there is no need to manually move inputs and outputs from one algorithm to the other.

Conclusion and Future Work

The issue of reproducibility is far too simple to be an issue of this magnitude. 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.

In the future, both AlgoRun and AlgoPiper could be expanded on in several ways:

- AlgoPiper could handle multiple inputs and outputs
- AlgoPiper could output graphics (eg. graphs, pictures)
- 2 MORE SUGGESTIONS
- ON ALGORUN (Maybe)

References

[1] E. Barnhart et al. (2015). PNAS 112(16):5045-5050. ← TEMPLATE

Acknowledgements

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

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.

DRAWING OF PIPELINE EXAMPLES

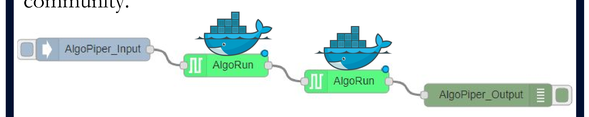
Despite the usefulness of pipelines, there are some complications when it comes to building pipelines. Reproducing pipelines is especially difficult because of the complexity of these pipelines. In fact, a study was done in which some scientists attempted to reproduce the results of a study, and it took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

As a result, we want an easy systematic way to share these 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 () and Bioboxes (). 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 container-based algorithm packaging solution AlgoRun [], and with newly developed, node-red based, tool, AlgoPiper [], we will show how to easily create reproducible and shareable pipelines for the computational biology community.

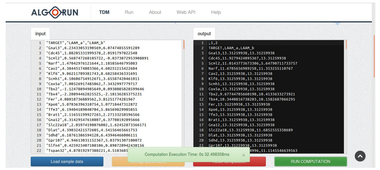


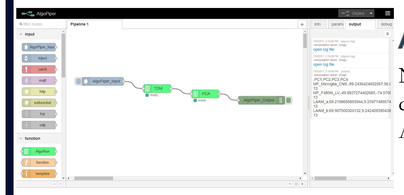
Methods

ALGORUN

Docker 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. 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





ALGOPIPER

Node Red is a graphical tool for wiring together the internet of things. It uses a drag and drop interface to AlgoPiper use Node Red in order to pipeline AlgoRun packages together. AlgoRun packages are nodes which AlgoPiper can connect together, creating a pipeline.

Application Example

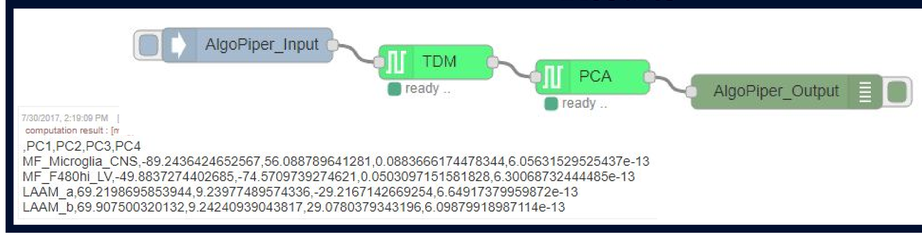
This example pipeline performs cross-platform comparison of gene-expression patterns in tissue-resident in-vivo macrophages. Macrophages are a type of white blood cell that "eats" menacing cells in an organism's body. There are several different types of macrophages, and they have different characteristics. In addition, macrophages that reside in different tissues can have different characteristics, and can, therefore, be different types of macrophages.

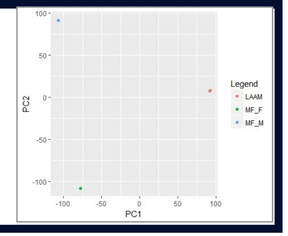
With the use of RNA-Seq and microarray technologies, scientists have studied gene-expression patterns in many cells. Currently, RNA-Seq is better at identifying gene-expression, but there is a plethora of microarray data that is available. To make RNA-Seq data compatible with microarray data, a new tool, Training Distribution Matching (TDM), was proposed. TDM converts RNA-Seq data to microarray data, so that they can be used together.

Principal Component Analysis is another algorithm that reduces the dimensions of data and can be used to represent multiple dimensions. This relatively simple algorithm can be used to classify and indicate differences in cells.

In this case study we will show how 3 macrophages from 3 different tissues in mice, the lungs, the central nervous system (CNS), and the liver, are all different. The macrophages that live in the lungs, named Lung Airways Associated Macrophages (LAAM), are newly discovered from our collaborator, Dr. Khanna's, research group. The other macrophages that live in the CNS and the liver are Microglia and F4/80 macrophages respectively.

Results





Discussion

As seen, it is extremely easy to package and pipeline shareable algorithms.

AlgoRun and AlgoPiper are both useful tools, however AlgoPiper could benefit from some improvements. 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. 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] E. Barnhart et al. (2015). PNAS 112(16):5045-5050. ← TEMPLATE

Acknowledgements

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

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:
Pipelines allow for _complex worflows and stuff_. Reproducing and sharing pipelines in computational biology is extremely difficult and time-consuming. Due to the complexity of computational workflows, it requires 280 hours to reproduce the results of an _experiment_. Computational workflows are complex because they have dozens of dependencies and run differently on varying execution platforms. As a result, virtual machines have been used to maintain a uniform execution platform. However, software containers, such as Docker, are emerging tools that provide the same benefits as virtual machines, but with faster execution times, and smaller images. Several tools already exist that allow for pipelining of algorithms and container orchestration such as Nextflow () and Bioboxes (). 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:
To correct this issue, we created AlgoRun () and AlgoPiper (). With a web user interface, it is simpler for users inexperienced in the command line, and with straightforward implementation, involving one extra file, it is easy to grasp. With the use of these two newly proposed tools, sheablility and reproducibility of pipelines is greatly increaased. In the case study, a pipline is created for cross-platform gene expression analysis, further illustrating the _simplicity_ of AlgoRun.

Methods

Docker Containers

Docker containers are an emerging tool to add portability to algorithms. Without a kernel, Docker containers are lighter and faster than virtual machines, the tool that is currently used. As a result, software containers are a better alternative to virtual machines.

AlgoRun & AlgoPiper

AlgoRun improves upon Docker and adds more modularity and features, while AlgoPiper allows for pipelining of AlgoRun packages. AlgoRun has a web user interface and runs online, allowing for remote execution of algorithms. It's user interface also makes it easier for users without as much technical expertise to operate. Because it uses a Docker container, AlgoRun also allows for portability of algorithms. AlgoRun's use of the RESTful API allows it to be easier to work with complex workflows and more modular. AlgoPiper adds to AlgoRun by allowing AlgoRun packages to be pipelined together. AlgoPiper also has a user interface and can be run online. In addition, AlgoPiper can also utilize the RESTful API and make it easier to implement into larger softwares.

Results

CASE STUDY HERE

- Both AlgoRun packages separately run
TDM and PCA (respectively) running on AlgoRun

- AlgoPiper Pipeline
Cross-Platform Gene Expression Analysis Pipeline in AlgoPiper

- Some words on why this pipeline is important
BlahBlah

- Some words on the data that is used
BlahBlah

Discussion

- As seen in the results, it is extremely easy to package and pipeline shareable algorithms.
- There is no need to run the algorithms separately and manually transfer the inputs and outputs to each other
- This will make advancement in this field significantly faster because scientists are not spending as much time and resources on reproduction of results.
- Sharing data will be very easy and it is simple to build off of each other's algorithms

Conclusion and Future Work

- In the future
  - AlgoPiper could implement multiple inputs and outputs
  - AlgoPiper could output graphics
- A simple issue such as this should not be causing a problem this large
- With AlgoRun and AlgoPiper, it will be much easier to fix this error
- It will not cost researchers so much time to simply reproduce an algorithm

References

[1] Wang, H. E., et al. (2014). Frontiers in Neuroscience
[2] Haury, A.-C., et al. (2012). BMC Systems Biology
[3] Huynh-Thu, V. A., et al. (2010). PLoS ONE
[4] Bradley, A. P. (1997). Pattern Recognition
[5] Gallo, C.A., et al. (2015). Briefings in Bioinformatics
[6] Vera-Licona, P., et al. (2014). BMC Systems Biology
[7] Marbach, D., et al. (2012). Nature Methods

Acknowledgements

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