Enhancing Reproducibility in Immunogenetics: Leveraging Containerization Technology for Bioinformatics Workflows

Rayo Suseno, Kristen J. Wade, Wesley M. Marin, Jill A. Hollenbach
University of California, San Francisco Department of Neurology

Introduction

Bioinformatics is experiencing a crisis of reproducibility, which inhibits research progress and calls into question scientific findings derived from unreproducible computational methods. Multiple evaluations on the reproducibility of bioinformatics workflows showcase the crisis we are experiencing, as only about 14% of them successfully ran to completion. Leveraging containerization technology has emerged as a promising solution to address these issues and streamline the deployment of bioinformatics workflows.

The field of immunogenetics research is especially in need of such workflows, as the high levels of genomic complexity found within immune loci often require the development of specialized and unique tools. For instance, our team published a pipeline called Pushing Immunogenetics to the Next Generation (PING), designed to genotype the KIR genes from short read data. **While PING has garnered considerable attention among immunogenetic researchers, investigators encountered numerous challenges in configuring the necessary software environment for PING.**

To that end, we aim to share our attempts on increasing the accessibility and reproducibility of immunogenetics pipeline. Using a container platform called Singularity, we showcase two pivotal pipelines that provides reliable high throughput analysis of large datasets not otherwise accessible with currently available tools: PING and MHConstructor.

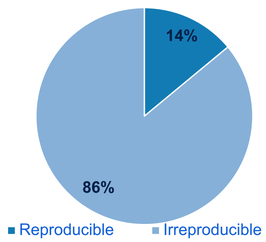

Figure 1

Previous attempts to assess the reproducibility of various bioinformatics workflows. Three studies performed independent assessment which found the following reproducible rates: 11%, 6%, and 26%, while the remaining workflows were irreproducible.

Containers

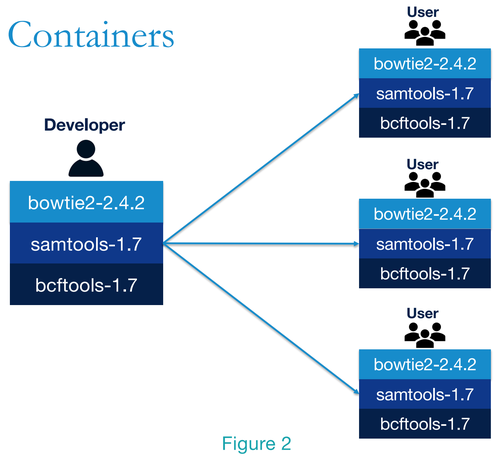

Figure 2

Container is a collection of packages and its dependencies that is necessary for a pipeline or workflow to run in any environment. The process of containerization ensure that the developer and user are using the same version of tools to promote reproducibility.

Benefits

1. **Ease-of-use for users.** Rather than installing their own dependencies, containers ensure that every component needed for the pipeline to run is taken care of. This is especially helpful for users that may not necessarily have a computational background.
2. **No reprogramming.** An existing pipeline can easily be containerized without having to rewrite the source code. Containers would simply run the same analysis using packages that adhere in the container, instead of the ones that are installed locally.
3. **HPC-friendly.** As opposed to a more popular platform like Docker, Singularity does not mandate its user to run the container using admin privileges. This feature offers flexibility when it comes to working with HPCs that are typically used in research universities.

Methods

Building a container starts with identifying packages and system requirements that are needed to run the pipeline. We started by noting down which version of our packages have been used during development. We also took note of system libraries that are needed to install different bioinformatics pipeline. For instance, *libtbb-dev* is mandatory for Bowtie2 to run. Therefore, the container is going to need *libtbb-dev* before it starts installing Bowtie2.

All these information is then written down in a Singularity definition file, which is used to build the container. Building the container is a one-line command in Singularity which creates a single-image file (SIF). Once built, both the input to the pipeline and SIF are used as arguments to execute your pipeline through the container from start to finish.

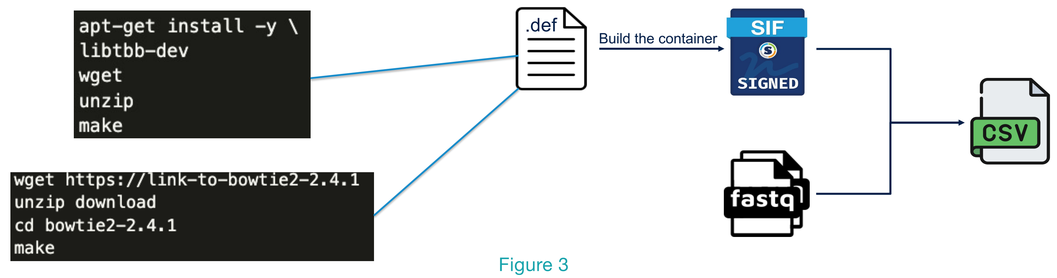

Figure 3

Building a container starts from a definition file that prescribes the necessary packages. The recipe then can be built to create an image SIF file which is used to run the pipeline alongside with the input files to produce the desired output.

Issue

One challenge that we encountered while developing MHConstructor is the need to run the pipeline in multiple Python versions. Specifically, Python 2.7 and Python 3.5 were needed in different steps of the pipeline due to its reliance on two bioinformatics tools: AMOS and RagTag. To address this, we turned to conda, a virtual environment manager. By building two different conda environments inside the container, we can seamlessly transition from one environment to the other, allowing us to use both AMOS and RagTag accordingly in our pipeline.

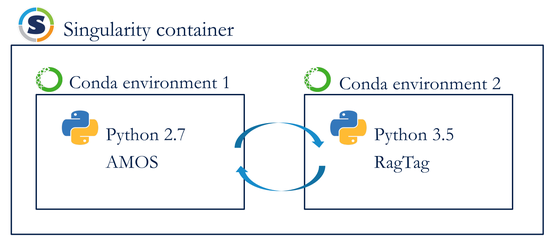

Figure 4

A graphical interpretation on how multiple conda environments can be stored within a Singularity container.

Containerized Pipelines

PING (Pushing Immunogenetics to the Next Generation) is a genotyping tool for the killer-cell immunoglobulin-like receptor (KIR) region from short-read paired-end whole genome or exome datasets. It is designed to detect all known and any novel KIR SNP variants.

MHConstructor is a short read *de novo* assembler for the MHC region, the most polymorphic region in the human genome. MHConstructor takes in short-read targeted, in-house whole genome, or 1000 Genome sequencing data to create an assembly that can be further used for association studies.

@hollenbachlab