# EVALUATING VIRTUALIZATION AND WORKFLOW MANAGEMENT SYSTEMS FOR BIOMEDICAL APPLICATIONS TO INCREASE COMPUTATIONAL SCIENTIFIC REPRODUCIBILITY

**By**

**Jason Hwee**

**A THESIS**

Presented to the Department of Medical Informatics & Clinical Epidemiology
and the Oregon Health & Science University
School of Medicine
in partial fulfillment of
requirements for the degree of

Master of Biomedical Informatics
September 2015

School of Medicine

Oregon Health & Science University

---

CERTIFICATE OF APPROVAL

This is to certify that the Master's Capstone Project of

Jason Hwee

*"Evaluating Virtualization and Workflow Management Systems for Biomedical Applications to Increase Computational Scientific Reproducibility"*

Has been approved

Shannon McWeeney, PhD

## Introduction

Success as a biomedical scientist depends on publishing findings that are relevant towards answering a biological question. A scientist needs to publish as frequently as possible in as many prestigious outlets as they can in order to meet certain requirements set forth by the institution in order to stay afloat in the hypercompetitive environment for obtaining grant funding. When research results are collected towards the end of a study or clinical trial there is a tendency for publication bias [8]. Publication bias is the incentivized act of the author favoring to publish positive results rather than neutral or negative results. Scientists are incentivized to publish positive results or the exciting findings and are encouraged to be innovative by pushing out new boundaries within the field. However, publication bias can be detrimental towards the reproducibility and replicability of a study as well as disrupt the self-correcting mechanism of the scientific method. Publication bias can lead towards a lack of transparency within the study where an unpublished neutral or negative result could be necessary for another scientist to reproduce and extend the original idea. In addition, without this strong foundation in scientific results, it may lead to concern among scientists that the original result could be false [4]. The negative results of another result project provides an essential learning experience for others by correcting the original methods thus leading to a more rigorous study design [3]. In a 2011 analysis conducted on the ClinicalTrials.gov databank, it was found that of the 76% of trials which had publications describing their own trial results, 74% of these were positive, 21% were neutral, and only 4% were negative or considered harmful [8]. Of the unpublished studies where the investigator responded, 43% resulted in a positive result, 57% were neutral result, and 0% were negative results. This bias has contributed towards an uncomfortable trend within all scientific communities; that all novel, exciting, and innovative findings deserve more emphasis over reproducibility of the experiments or the methods.

## Reproducibility, Replicability, and Rigor

Reproducibility is fairly simple in a scientific sense and asks the following question. Can another scientist take the data and findings produced in study A, run the full analysis according to the original published methodology again in study B, and reproduce the original result? Reproducing a result or an entire study is conducted under the false assumption that the scientist has access to the original publication and the original data set during the original analysis. Reproducibility of a study is so incredibly important in all of science because it sets the foundation for confidence in future results.
Replication is also an important factor which contributes towards the reproducibility of a study. It asks, are the methods and experiments rigorously and exhaustively designed so that the original group conducting the science can replicate their own results with their own equipment? An experiment that takes multiple attempts to obtain the same result can be said that it is difficult to replicate [9].

Each laboratory is conducting research they believe is providing important contributions in their respective fields and they each are operating under the belief that they are

conducting science upon a foundation that is sound and wholly based on previous evidence. Studies published in high impact journals such as Nature or Science should theoretically be experimentally sound and are perceived to be highly influential in their respective fields. These influential studies are cited time and time again by additional groups each attempting to build and extend upon the same innovating idea [9]. Scientific reproducibility is not a new concept among scientists nor has it been completely ignored. This literature only means to garner attention to the idea that the scientific community must take steps towards keeping the focus on rigorous experimental design and publishing reproducible research [16]. In an analysis conducted by a team of Amgen scientists led by C. Glenn Begely, a former employee of Amgen, found that 47 of 53 identified “landmark” publications from preclinical trials could not be replicated [2]. Only 6 (11%) of the 53, landmark studies for oncology drug target projects were able to be replicated. It was often the case that investigators only presented the results of a single experiment or declared that specific experiments may not be representative of the entire data set however supported their underlying hypothesis [1]. Another analysis conducted internally by a team at Bayer, a german multi-national chemical and pharmaceutical company, collected data from 67 of their in-house projects where 70% of them were oncology related. They received input from 23 scientists, who are also heads of their departments, and found that only ~20-25% of in-house project findings were consistent with the published data [6].

**Computationally Oriented**

Computational science has opened new and exciting research opportunities in biomedical fields, however the nature of the work has revealed limitations in our ability to evaluate published findings [5]. Advancements made in computational methodologies, new technologies, or simply increased computing power have contributed to our fast growing ability to collect data. In addition, with the use of the internet, scientists are able to collaborate from around the globe in consortium guided research to develop large combined data sets that are both complex and high-dimensional. The availability of large datasets housed in public databases has allowed scientists from computationally focused fields to engage directly in biomedical research.

Reproducibility in computationally driven research can be a non-trivial task especially when dealing with complex high-dimensional data sets. Analysis of the results can require computational resources that may otherwise not be available to the independent researcher. The researcher may also not be sufficiently skilled to manipulate multiple software packages, command line interfaces, operating systems, or the computational resources themselves. In addition, there may be important nuances in the implementation of such software pipelines which may include the pre-processing or manipulation of data. Some of these important nuances can come from programming scripts developed specifically in-house to increase efficiency within the original laboratory however are not deemed important enough to be included or shared in the final published work.

**The Reproducibility Spectrum**

In computational science, researchers require at least a minimum standard for attaining reproducibility of a publication. This reproducibility standard requires that the data and the code used to analyze the data be made available and accessible [5]. The aim of requiring a reproducibility standard among computational researchers is to bridge the gap between a non-reproducible work, which consists of just the publication itself, and fully reproducible work, which consists of the publication and all of the necessary code in an executable format. Between being non-reproducible and fully reproducible there is a spectrum of partially reproducible work which depends on which data and code are made available. For scientists conducting computational research, it is essential that the focus is not on the final figures , the concluding remarks, or aesthetically pleasing graphs, but the hope that evaluating such complicated methodologies can be assisted by transferrable code [15].

**Virtualization**

In theory, a computational software pipeline can have a detailed log of every action taken by the original analysis. In practice however, this may require the assistance and expertise of a virtualization or workflow management system. Computational virtual environments are a self-contained environment that provides all of the necessary resources required for a specific project or task. Virtualization can be done on many levels which include virtualizing computer hardware, operating systems, or programming environments.  A virtualized environment can have many benefits over dedicated hardware access for both the user and the administrator of the computational resource. In computational oriented fields, data sets are large, complex, and highly dimensional. Many of these studies require the use of algorithms in machine learning, statistical methodologies, and signal processing all of which require computing resources that may not be available to all scientists [5]. Many researchers who wish to conduct or reproduce a result will have to rely on shared institutional resources which can provide computational power on orders of magnitude higher than a personal computer. In general, a shared computational resource is administrated by a management group which is tasked with maintaining, monitoring, and updating the system.

Each individual will have their own project requirements that can differ greatly from the next. Some may require the use of the latest up to date software packages whereas others may require installation of older deprecated which may no longer be maintained. As a user of the system, permissions will be in a restricted mode where they will only be able to read, write, and execute programs that they own. A user will not be able to change, install, or modify the system in any way in order to maintain the integrity of the system as a whole. As such, a user will explicitly have to ask administrators for additions or changes to the system in such the case where a new software package is to be installed. In general, multiple software packages, dependencies, and access to the same computational resources can create a logistical nightmare for both the users and the administrators. The solution is to implement virtualization software to create virtualized environments for each user where the ability to install, configure, modify, and execute any software packages is allowed and has minimal impact on the resource as a whole.

Virtualization software can not only provide a specific level of autonomy for the individual user but it can also be used to configure and share entire virtualized environments across multiple computers. For instance, for the sake of scientific reproducibility an author can decide to create a pre-configured virtualized environment that contains the software, scripts, and settings necessary for a specific analysis. Then, instead of trying to replicate every step of the analysis as described in the publication, the independent group can download the shared virtualized environment and implement the analysis from the authors point of view. Ultimately, virtualized environments are important in reducing the number of external variables that an independent group needs to control for when attempting to reproduce a published computational finding.

**Workflow Management System**

A workflow is a collection of activities systematically organized into a structured series of repeatable steps. Scientific workflows are highly valued in bioinformatics because of its ability to interconnect tools from across multiple disciplines and because of the need to create consistency in processing multiple large data sets [10].
Workflows can be representation of how real work is processed manually by people at a specific organization which can then be used as a template of instructions for others to follow. A common goal in developing a workflow is to keep the processing steps consistent across multiple iterations. In addition, workflows can essentially create detailed logs of every action taken by the user in case it needs to be examined.

**Purpose**

The primary objective of this document is to provide a resource for individuals who have no prior experience in computer science and are looking to conduct genomic computational experiments involving statistical analysis. In addition, this document also serves to provide a brief assessment for biologists or bioinformaticists on the usability of Docker virtualization and workflow management systems like Galaxy and the Taverna suite.

**User Assessment**

Implementing a new scientific workflow is not a trivial task and there may be many reasons for wanting to do so. For example, the user may want to test out a new protocol for their lab in order to see which is more efficient or they may want to increase lab efficiency in analysis by automation. They may also want to try and ensure reproducibility within their lab by creating a documented series of steps that is useable between different technicians or analysts.

It is apparent that our need for computation in biology is increasing and will become more mainstream in universities and laboratories with little extraneous effort. Contributions to bioinformatics will come from not only those who use bioinformatics tools but those who analyze, develop, and implement computational methodologies. We must consider the multidisciplinary nature of scientists who desire to apply

bioinformatics analysis in their research. What is the overall skill level of the scientist? Is the user a biologist or rather a bioinformaticist? A skilled multidisciplinary scientist must draw from their experience across all scientific fields to find solutions to a problem that may otherwise prove too difficult. A biologist who would like to implement a scientific workflow may need to address specific questions before pursuing a computational solution. Such as, what is the technical skill level of the user? A biologist would want to be comfortable working with computer systems. A basic understanding of computer components like the functionality of a central processing unit (CPU) or how random access memory (RAM) is needed and its role in the algorithms being used. A shared resource like a university computer cluster would likely provide a specific number of CPUs and RAM per user per job to be sure that a user does not occupy all of the resources. If the server resources are not automatically managed then it is up to the user to understand how to allocate computational resources without adversely affecting other users.

Another basic skills consideration for the biologist is whether or not the user is comfortable with working in a Unix environment. Is the user comfortable working in a Unix like environment? A Unix based operating system has several important attributes that make it a predominant environment among academic research institutions. Attributes such as being multi-user friendly, multi-tasking, and being highly configurable and distributable make Unix based operating systems an ideal environment for research. Multi-user refers to the ability of the operating system to have more than one active use at any given time. The modular design of a Unix based operating system allows each user to have their own working space. Each user is responsible for their own files and are generally not allowed to edit other user files without their permission. Multi-tasking or multi-processing refers to the ability of the operating system to gain access to critical system resources in order to run multiple core processes. Unix-like operating systems such as Linux, are highly configurable by the system administrators to suit the needs of the users. Administrators may install tools that allow the computing cluster to automatically allocate resources to users such as CPU or RAM. Other administrative tools can allow files to be distributed across multiple locations in an effort to produce efficient indexing for lookup, secured backup capabilities, and overall system maintenance and longevity.

To take advantage of these capabilities, computational clusters often do not have a graphical user interface (GUI) but rather use a text-based command line interface (CLI). The CLI is a text-based way to access core programs that are particularly suited for automation and delayed programming tasks. The CLI can be considered as a way to programmatically automate repetitive tasks or manipulate large text files. It can grant the user much more control over the file system as well as allowing the user to automatically string together various steps in a workflow.

The CLI is unlike GUIs in that it requires an abstract level of thinking about computer programs that many users may not be familiar to. According to web analytics that monitor operating system market shares, it can be observed that more than 50% of worldwide users using personal computers (PCs) run Windows as their day to day

operating system and around 5% of users are running Mac OSX 10 [18]. Both Windows and OSX operating systems have GUIs that are the primary component of the user experience. Ultimately, it suggests that users are more familiar having graphical elements that they can directly manipulate giving rise to the perception that CLIs require a steep learning curve.

**Docker Virtualization**

The Docker software is a command line platform which can be used to run, execute, and distribute virtualized environments in the form of system images and containers. Pre-configured images and containers for a wide variety of purposes are maintained by the community in an open source fashion and are made available through the Docker internal distribution system called the registry. Docker can be used as a platform for hosting many common applications such as a virtualized R server, a Galaxy server, or even a simple virtualized file distribution web server [12]. Docker can be used as a safe way to allow multiple users access to a shared resource. While it is true that users can access the same computer cluster, it can also be a hassle from an administrative standpoint, to manage user permissions or manage a continual chain of system requests. Each user will have their own specific requirements as to which software packages they need for their specific project.

**Use case – chemiinformatics**

A user may want to replicate the analysis of a recently published chemical informatics study. This study utilizes a chemi-informatics pipeline that consists of in-house pre-processing scripts as well as domain specific software packages. This may involve installation of older software packages that were in widespread use 6 months ago for multiple steps along the pipeline.

First in the effort of replicating this study, the user may find that a module called PerlChemBio 1.4 is required in order to proceed further and does not exist on the system in any form. In turn, the user contacts a responsible administrator and requests that the module be installed so that it can be used. There is a standing request to install PerlChemBio because it has been addressed before by multiple users in the department as a shared resource so the administrator accepts the request and installs the software package system wide. The administrator proceeds with the installation and notifies the requesting user. However, the administrator believes it is best to install PerlChemBio 3.0 which is the most recent iteration of the module with significant changes to the many of the algorithms. In order to replicate the original result, the user can request that the administrator install an older version of PerlChemBio. However unbeknownst to the user, installing an older version of the module will conflict with the new version of PerlChemBio 3.0 and therefore negatively affect the remaining users.

In an ideal situation, the user would be able to install PerlChemBio version 1.4 without it negatively affecting any other user. The user could continue to install multiple software packages with different dependencies without worrying about system-wide conflicts. The

solution to this problem would be to give the user a sandbox environment where they have full control over the installation processes without it conflicting anyone else. This would provide the user increased flexibility and allow the administrator to have better control of the system on a much high scope.

**Use case - R programming environment for statistical computing**

As stated earlier, Docker can be used to create a virtualized R container environment. R is a programming language used for statistical computing and graphics [20]. It is popular among researchers because it is relatively easy to install on your local machine allowing the first time user to learn how to program. In many cases, it can be common to perform all of the necessary processing steps on the computational cluster. After the output files are produced, you can then transfer the results to a local machine in order to perform analysis in R. However, there are some cases where it more useful to use an R environment that is hosted on the same computational cluster. The first reason being that running an analysis can also require a greater computational resource not usually found in a local machine. It may be necessary to utilize the shared computational power of the cluster in order to perform an analysis requires machine learning.

The second reason is that datasets are growing in size as well as length. It is being common practice to create joint datasets with information collected from various groups around the world. In an ideal situation, the resulting files would stay in one location such as the computer cluster without having to be transferred to multiple locations. It would be beneficial for any users who want to perform analysis on these files if there could be a hosted R environment in which to work in. The R environment would have the same capabilities as a local installation except it would have access to greater resources. Setting up an R server on a shared resource is a task normally reserved for administrators of the cluster. There are many reasons as to why administrators may not allow a dedicated R server process to run indefinitely on their system. One such reason is related to uncontrolled resource management of the system as a whole or there may be reason to suspect a security vulnerability within hosting an R server. With enough support from the users, they may be able to create a case for bringing a beneficial new service online for the community. In this situation, a layer that the users could control such as Docker would be useful. Docker would act as the layer between system resources and permissions while also allowing the users of the cluster to install their own personal container of R. It is possible for a single user to set up R version 2.0 and provide it as a stable resource for the community. However, Docker will also allow users to install their container with R version 3.0 if they so desire it. Many users will have different requirements depending on their project so it is reasonable to expect those users to want as much customization as possible without affecting the system. The advantages of using the Docker software for testing and sharing virtualized environments allows for the user to fully have control over their working environment while still staying in the confines of a containerized sandbox.

**Assessment of Docker**

Installation of Docker took place on a local Ubuntu 14.04 machine for initial evaluations. Like any software, Docker installation requires administrative permission in order to be used. Those using a popular Linux distribution like Ubuntu and without expert knowledge of the command line can download the necessary Docker binaries and install by executing the startup script. Another option is to download Docker through the available package management system such as the Advanced Package Tool (apt). Docker is also available cross-platform for Windows and OSX operating systems. Once Docker is installed, the user can utilize the internal Docker software distribution system called the registry in order to download a pre-configured image or container that suits your needs.

**Usability**

The Docker virtualization software can be considered fairly straightforward and very easy to use for both biologist or bioinformaticist. In the common case, Docker will be installed on a shared institutional resource by an administrator who will be able to grant access to the Docker group. A biologist without computational expertise would be able to access the Docker services through the command line with minimal understanding of the virtualization process or the file structure for the Linux operating system. From here, a biologist can perform the main functionality of Docker which is installation, starting, and stopping of images or containers.

For example, if the user wanted to utilize the Galaxy framework in order to conduct a gene expression analysis on their current research then they could do so easily. A user can easily navigate to the Docker registry website located at https://hub.docker.com/explore and perform a search for Galaxy. Here, they will find multiple Galaxy framework images each customized by their original authors ready for download. Once an image is chosen, the user can return to the command line and issue a simple command in order to download a pre-configured image. From here, the user can issue another simple command to start the Galaxy image which will then be available as a container on the default ports. After the Galaxy process has started, the user can navigate to the Galaxy container through their web browser and begin using the workflow system as normal.

**Proficiency with the Shell**

Docker requires minimal knowledge and comfort of a Linux shell in order to perform many of the basic functionality. Most users will only have to worry about installation, starting, or stopping container processes. Basic knowledge of how to navigate the shell and understanding of basic shell commands will allow a user to use Docker within a few hours.

A bioinformaticist or computational biologist can benefit from their expertise and comfort with the shell in order to automate some of the Docker processes. From here, an advanced user can write shell scripts in order to automatically start and stop Docker processes when the session begins or the job is finished. Docker processes are persistent, meaning that the container will continue to run until it is closed.  It can also result in the

user unintentionally leaving a Docker process online all of the time. Automating this process can be useful in the case where the system resources are limited and such activity needs to be monitored.

**Flexibility and Customizability**

Docker allows the user to install and execute any number of containers for whatever virtualization need arises. The user can easily download a pre-configured Docker image and begin to take advantage of the relatively quick start up times. Docker also provides the user with the ability to create and configure their own container by downloading a base image. A base image serves as a starting point for configuring a new virtual environment from anew. From here, an advanced user who is proficient with manual setup of a Linux operating system can configure a virtual environment to contain any software packages as needed just like it were a standalone physical machine. This will allow a Bioinformaticist or computational biologist to create and share virtualized computing environments for specific tasks without having to worry about breaking software dependencies or setting user permissions.

Since the initial release of the Docker virtualization software as open source in 2013 there have been a steady rise in its popularity due to its clear advantages across all fields which require some form of computation. As of 2015, there are over 1000 contributors from all over the world with companies such as Redhat, IBM, and Google contributing as well [17]. Despite its fairly new release, Docker is showing promise in the scientific reproducibility in regards to helping scientists reconstruct complicated computational studies.

**Workflow management system RNAseq analysis use case**

High-throughput sequencing of mRNA (RNAseq) is a highly valuable assay that can provide scientists with the ability to discover new genes and transcripts [7]. In addition, RNAseq provides a measurement of the transcripts relative abundance which allows scientists to quantify important changes in transcript expression at a given moment in time [13]. RNAseq experiments generate an enormous amount of raw sequencing reads thus prompting scientists to employ sophisticated algorithms and computational methods, all of which require significant computational power. Processing the hundreds of gigabases per run is a constructed effort requiring many steps in data cleanup, alignment of the reads to the reference genome, transcript assembly, annotation, merging, differential expression analysis, and generation of plots. Each step calls for specific analysis tools, all of which are state of the art and are continuously being improved upon. At each step, large volumes of data are being manipulated and conjoined as they are subjected to rigorous transformation. From a reproducibility perspective, each step is a potential introduction in error. A user must carefully understand and purposefully select input and output variables for each algorithm. In addition, the user must keep track of the version of each piece of software in order to faithfully reproduce an RNAseq experiment faithfully.

## Manual RNAseq workflow implementation

The majority of users who are interested in implementing their own RNAseq pipeline for research will likely consider a manual installation of all related software packages. There are multiple steps in an RNAseq analysis pipeline which include alignment to the reference genome, transcript assembly, annotation, merging transcript assemblies, and finding differentially expressed genes. Finally, for visual inspection the user can use R programming language modules like CummeRBund to create plots of transcript abundance and differential expression results. In this RNAseq implementation, Tophat/Bowtie will be used for the alignment step. The Cufflinks suite of tools will be used for transcript assembly, merging, and identification of differentially expressed transcripts.

## Usability

A manual implementation can be advantageous because the user can maintain full control over the system environment and control all of the input variables for each one of the software steps. For each step in the protocol the user must consider specific versions of the software which can impact the outcome of the expression results. For example, the first step in the tuxedo suite protocol utilizes a splice junction mapper called Tophat which in turn utilizes a high-throughput short read aligner algorithm called Bowtie to align RNAseq reads to a reference genome. Both of these software packages need to be acquired separately and made sure that the are properly extracted in the user's home directory.

The usability assessment for the initial alignment step TopHat requires that the user work in the shell environment since there is no graphical user interface for the user to select input variables. Both a biologist and a bioinformaticist would be using the same exact commands to initiate the TopHat alignment procedure. Before the user proceeds, regardless of skill level they should understand the meaning of the parameters such as the number of processors they are willing to allow the algorithm to utilize. In addition, the user needs to provide the TopHat algorithm with a gene list as well as the location of the indexed reference genome. Once the user chooses their desired parameters, then it is relatively easy for a biologist or bioinformaticist to execute.

## Level of proficiency

A substantial level of computational proficiency is required to operate in the Linux shell in order to setup a proper RNAseq analysis environment. If a system-wide installation of TopHat and Bowtie are not available, then the user will have to obtain these from their respective sources.
Although previously stated that the usability of the algorithms was simple enough to execute, the user may run into a number of issues that need to be troubleshooted. A common issue the user may run into is that sometimes the location of the Bowtie aligner is unknown to the TopHat software.

**Flexibility and customizability**

Various changes in the software can affect the expression results differently depending on how the authors maintain the software. For instance, the authors may choose to change the way a user inputs a variable when running the algorithm.

**Galaxy Workflow management system**

The Galaxy project is a free, web-based platform used for data driven biomedical research. In essence, it aims to provide a graphical web interface so that users without advanced computational training can still utilize the same algorithms as a trained computational biologist or bioinformatician [14]. Galaxy attempts to lower the barrier of entry so that a first time user can visually construct a biomedical research pipeline inside a web-browser without learning how to use a new unfamiliar operating system or CLI.

**Usability**

Galaxy is intended to provide an easy to use web environment in which a user can run popular bioinformatics tools. A powerful feature of Galaxy is that the user environment is independent from the operating system on which it runs. Users that are familiar with Windows and OSX will be able to use Galaxy through a series of graphical menus which allow for a wide range of input variables. Users do not have to download and install their software packages independently but rather through Galaxy's toolshed feature. The Galaxy toolshed is a built in tool that links Galaxy directly to community supported software repositories. This allows the user to easily search for and download any algorithm, analysis tool, or software that is required with the click of a button. Once the tools are downloaded, users can upload their raw data via file transfer protocol (FTP) into Galaxy using the 'Get Data' menu.

**Level of proficiency**

Another powerful capability Galaxy provides is the workflow feature that allows the user to string together a series of tools. In order to create a Galaxy workflow, the user must specify input and output files at each step as well as the directionality of the step. In addition, the user must also specify beforehand all of the algorithm settings. For example, if the user wanted to create the first step of an RNAseq analysis workflow for differential gene expression analysis then they simply need to open the Galaxy workflow editor. The user can choose to use TopHat/Bowtie to align reads to the reference genome. By clicking on TopHat/Bowtie, Galaxy will automatically produce a graphical object that represents the first node in the workflow. The user can then choose the input file format and then set the algorithm variables in the side menu. Once the second node is created, the user can then chain the two nodes together by dragging arrows from the output node of TopHat to the input node of the Cufflinks suite. By organizing things in a graphical user interface, the user can easily manipulate the algorithm parameters and execute the procedures without having to worry about

**Flexibility and customizability**

There are however, a number of limitations to Galaxy which may need to be taken into consideration when deciding whether or not it is appropriate for the scientific need. Galaxy hosting is available in 3 primary distributions, the first form is a publically hosted Galaxy. There are many public servers hosting Galaxy worldwide with an innumerable number of contributors. Galaxy hosted on a public server is an accessible way to become familiar with the Galaxy platform without having to personally install any overhead. However, there is no way to guarantee data privacy while conducting research on a publically managed server. This could affect how and when authors decide to publish or could affect patient confidentially by inadvertently uploading health information.
The second form for Galaxy is a cloud based privately hosted server that aims to provide a paid for computational Galaxy environment. The cloud based Galaxy can be expensive to rent and still does not guarantee data privacy. In addition, data analyzed within the cloud will need to be uploaded via the internet via FTP. Transferring hundreds of gigabytes over the internet into Galaxy is slow and will take numerous man hours of waiting.

The third form of Galaxy is a local installation most likely on a shared computational resource provided by the institution or on a personal computer. Installation of Galaxy server requires administrative access to the computational cluster as well as moderate familiarity of the CLI. Once installed, the administrator will want to set up user accounts and permissions as well as create a disk space partition for users to upload their data. The Galaxy Toolshed, as helpful as it may be, can be cause for user confusion. It is relatively easy to install a software package, algorithm, or tool of the users choice. However, it is not always possible to install the exact version the user desires. For example, the TopHat2 software is available on the provided Galaxy Toolshed repositories however there are multiple versions maintained by different groups. Each group provides a date and the version of TopHat2 however none provide the absolute latest version. In addition, there is no availability for ‘inbetween’ versions or for selection among ‘all past versions’ of TopHat2. Updates to the algorithm available are largely held responsible for by whichever group is updating their repository.

Galaxy as a web-based platform can be very helpful in conducting reproducible work. Using it, scientists can share tools and publish workflows that they themselves have used to publish their results. Anyone can learn to use Galaxy, regardless of their proficiency in shell programming.

**Taverna workflow management system/suite**

Taverna is an open source workflow management system that is designed to integrate distributed web services or local tools into a streamlined analysis pipeline. These workflows can be executed locally or on a more powerful computer like distributed institutional computing cluster. Taverna is a suite of tools that comprises of Taverna Workbench, Taverna Server, and a community supported external repository of tools and bioinformatics data sources. The Taverna Workbench is available for download on

Windows, OSX, and Linux operating systems allowing for all users the ability create workflows.

**Usability**

The Taverna suite of tools is a highly sophisticated workflow management system. Taverna primarily relies on bringing together tools in the form of web services using a representational state transfer (REST) architecture style for networked applications. In its simplest form, Taverna sends a request to an external web service which in turn sends a response back to Taverna. The web service can be anything from a country-city database, gene lookup in a KEGG pathway, or a basic local alignment search (BLAST).
Work flows are incredibly customizable in Taverna and can utilize programmatic functions like loops to create sub workflows. This can be advantageous for the user because workflows can share some of the same components allowing for less redundancy. For a new user, the Taverna workbench can be daunting and would require a significant time investment in order to learn just the basic functionality of running a sample workflow.

**Flexibility and customizability**

Taverna is an extremely customizable workflow management system that incorporates powerful programmatic functionality in order to integrate tools across any domains with fine input/output control. However, Taverna requires a considerable time investment in both the learning curve as well as workflow construction. Since Taverna is domain independent, biologists looking to implement workflows in their labs may need to have a basic understanding of computing concepts like input/output (IO) variables and programming control structures like loops. In addition, the primary advantage to using Taverna is the distributed nature of the tools that can be incorporated into a workflow. While the tools can be hosted and distributed on any server, it requires a considerable amount of programming for the hosting institution to create a REST like application programming interface (API). This may be trivial for simple tools that have a single input and output but may become more complicated with much larger data sets.

The customizability of the Taverna suite is directly related to the users understanding of distributed services and programming experience. If the user is a biologist with little programming experience, then they will be limited to utilizing the tools that are already publicly available for use. For example, there are many tools in the Biocatelogue registry in which a user can incorporate into their workflow however these are relatively simple tools with minimal input and output data. If the user wanted to implement an RNAseq analysis, they will be hard pressed to find a publicly hosted service for performing a raw reads alignment or for transcript assembly due to the high computational requirements.

**Taverna2Galaxy**

Taverna2Galaxy is an effort to create interoperability between Galaxy and Taverna workflows due to the large computational requirement of RNAseq analysis tools and the

advanced level of skill proficiency required. Creating and hosting tools that can be utilized in an RNAseq experiment requires significant time in creating a web service layer and significant publicly available computational resources. The time investment an independent group would take to develop this service layer may be prohibitive due to the limited number of people involved. Galaxy is an existing framework for installing and executing high throughput biomedical data which is also already publicly available. The Taverna2Galaxy effort would provide powerful workflow automation coupled with the high-throughput tools that are already available through the Galaxy framework [11]. Taverna is a promising workflow management system that has a extremely high level of capability and functionality. Because of these features, Taverna has recently been taken over as an Apache Incubator project to further develop the workflow automation technology.

**Concluding remarks**

Scientific discovery is increasingly being influenced by the achievements in computational research. Analysis of distributed and highly-dimensional data sets require the use of complex statistical models and powerful algorithms which may be comprised of hundreds of steps developed by multiple groups around the world [19]. Piecing together and managing these steps a scientific workflow presents many challenges and requires a great deal of effort from many contributors. Scientific workflows have emerged as a promising system to represent the management of complex components that comprises a fully functional pipeline. These workflows should also allow for a high degree of flexibility, by supporting the researchers desire to perform modification or extension of existing workflow components. In addition, virtualization software such as Docker is a promising venture that allows scientists an additional avenue for sharing fully developed workflow environments therefore supporting the idea of scientific reproducibility. Virtualization technologies will allow scientists to implement and share various workflow components that traditionally require significant overhead. The ability to combine multiple data sets, computational tools, analyses is becoming a common practice among scientists. Individuals are no longer solely responsible for the data they generate or the tools they develop, for it is the responsibility of the community as a whole to enable the reproducibility of scientific findings so that we can evaluate others with confidence.

## References


1. Begley C, Ellis L. Drug development: Raise standards for preclinical cancer research. Nature. 2012;483(7391):531-533.
2. Begely S. In cancer science, many discoveries don't hold up [Internet]. Reuters. 2015 [cited 15 September 2015]. Available from: http://www.reuters.com/article/2012/03/28/us-science-cancer-idUSBRE82R12P20120328
3. Goodchild L. Why it's time to publish research failures [Internet]. 2015 [cited 24 September 2015]. Available from: http://www.elsevier.com/connect/scientists-we-want-your-negative-results-too
4. Ioannidis J. Why Most Published Research Findings Are False. Plos Med. 2005;2(8):e124.
5. Peng R. Reproducible Research in Computational Science. Science. 2011;334(6060):1226-1227.
6. Prinz F, Schlange T, Asadullah K. Believe it or not: how much can we rely on published data on potential drug targets?. Nature Reviews Drug Discovery. 2011;10(9):712-712.
7. Trapnell C, Roberts A, Goff L, Pertea G, Kim D, Kelley D et al. Differential gene and transcript expression analysis of RNA-seq experiments with TopHat and Cufflinks. Nat Protoc. 2012;7(3):562-578.
8. Vawdrey D, Hripcsak G. Publication bias in clinical trials of electronic health records. Journal of Biomedical Informatics. 2013;46(1):139-141.
9. Yaffe M. Reproducibility in science. Science Signaling. 2015;8(371):eg5-eg5.
10. Wolstencroft K, Haines R, Fellows D, Williams A, Withers D, Owen S et al. The Taverna workflow suite: designing and executing workflows of Web Services on the desktop, web or in the cloud. Nucleic Acids Research. 2013;41(W1):W557-W561.
11. Kouskoumvekaki I, Shublaq N, Brunak S. Facilitating the use of large-scale biological data and tools in the era of translational bioinformatics. Briefings in Bioinformatics. 2013;15(6):942-952.
12. Docs.docker.com. Get started with images [Internet]. 2015 [cited 17 September 2015]. Available from: https://docs.docker.com/userguide/dockerimages/
13. Trapnell C, Pachter L, Salzberg S. TopHat: discovering splice junctions with RNA-Seq. Bioinformatics. 2009;25(9):1105-1111.
14. Wiki.galaxyproject.org. Learn - Galaxy Wiki [Internet]. 2015 [cited 13 September 2015]. Available from: https://wiki.galaxyproject.org/Learn
15. Fomel S, Claerbout J. Guest Editors' Introduction: Reproducible Research. Comput Sci Eng. 2009;11(1):5-7.
16. Landis S, Amara S, Asadullah K, Austin C, Blumenstein R, Bradley E et al. A call for transparent reporting to optimize the predictive value of preclinical research. Nature. 2012;490(7419):187-191.
17. GitHub. docker/docker [Internet]. 2015 [cited 10 September 2015]. Available from: https://github.com/docker/docker
18. Netmarketshare.com. Operating system market share [Internet]. 2015 [cited 11 September 2015]. Available from: https://www.netmarketshare.com/operating-system-market-share.aspx?qprid=10&qpcustomd=0

19. Gil Y, Deelman E, Ellisman M, Fahringer T, Fox G, Gannon D et al. Examining the Challenges of Scientific Workflows. Computer. 2007;40(12):24-32.
20. R Core Team (2015). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL http://www.R-project.org/.