Check for updates

SOFTWARE TOOL ARTICLE

A guide and best practices for R/Bioconductor tool integration in Galaxy [version 1; peer review: 1 approved, 1 approved with reservations]

Nitesh Turaga¹, Mallory A. Freeberg¹, Dannon Baker¹, John Chilton², Galaxy Team, Anton Nekrutenko², James Taylor¹

¹Department of Biology, Johns Hopkins University, Baltimore, USA

²Department of Biochemistry and Molecular Biology, Pennsylvania State University, University Park, USA

V1 First published: 24 Nov 2016, 5:2757
<https://doi.org/10.12688/f1000research.9821.1>
 Latest published: 24 Nov 2016, 5:2757
<https://doi.org/10.12688/f1000research.9821.1>

Abstract

Galaxy provides a web-based platform for interactive, large-scale data analyses, which integrates bioinformatics tools written in a variety of languages. A substantial number of these tools are written in the R programming language, which enables powerful analysis and visualization of complex data. The Bioconductor Project provides access to these open source R tools and currently contains over 1200 R packages. While some R/Bioconductor tools are currently available in Galaxy, scientific research communities would benefit greatly if they were integrated on a larger scale. Tool development in Galaxy is an early entry point for Galaxy developers, biologists, and bioinformaticians, who want to make their work more accessible to a larger community of scientists. Here, we present a guide and best practices for R/Bioconductor tool integration into Galaxy. In addition, we introduce new functionalities to existing software that resolve dependency issues and semi-automate generation of tool integration components. With these improvements, novice and experienced developers can easily integrate R/Bioconductor tools into Galaxy to make their work more accessible to the scientific community.

Keywords

Interoperability, Bioconductor, R, Galaxy, Open Source, Bioinformatics

Open Peer Review

Approval Status

	1	2
version 1 24 Nov 2016	view	view
1. Paul A. Stewart , H. Lee Moffitt Cancer Center & Research Institute, Tampa, USA Dekai Rohlsen , H. Lee Moffitt Cancer Center & Research Institute, Tampa, USA		
2. Houtan Noushmehr , Henry Ford Hospital, Detroit, USA University of São Paulo, São Paulo, Brazil Tiago Silva , University of São Paulo, São Paulo, Brazil		

Any reports and responses or comments on the article can be found at the end of the article.

This article is included in the **Galaxy** gateway.

This article is included in the **Bioconductor** gateway.

This article is included in the **RPackage** gateway.

This article is included in the **Bioinformatics Education and Training Collection** collection.

Corresponding author: Mallory A. Freeberg (mallory.freeberg@gmail.com)

Competing interests: No competing interests were disclosed.

Grant information: This project is funded in part by the National Institutes of Health [U41 HG006620].
The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Copyright: © 2016 Turaga N *et al.* This is an open access article distributed under the terms of the **Creative Commons Attribution License**, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

How to cite this article: Turaga N, Freeberg MA, Baker D *et al.* **A guide and best practices for R/Bioconductor tool integration in Galaxy [version 1; peer review: 1 approved, 1 approved with reservations]** F1000Research 2016, 5:2757
<https://doi.org/10.12688/f1000research.9821.1>

First published: 24 Nov 2016, 5:2757 <https://doi.org/10.12688/f1000research.9821.1>

Introduction

The Bioconductor Project (<https://www.bioconductor.org/>) provides one of the largest suites of open source bioinformatics tools for analyzing genomics and biomedical data from diverse high-throughput assays, including DNA microarrays, flow cytometry, and deep sequencing^{1,2}. Bioconductor has an active user community of researchers from across a spectrum of biomedical professions, who routinely analyze complex genomics datasets. Bioconductor tools and packages are primarily based on R (<https://www.r-project.org/>), a programming language and integrated environment developed for statistical computing and data visualization³. By submitting R packages to Bioconductor, developers can easily distribute powerful statistical analysis and visualization tools, while supporting efforts towards reproducible scientific research.

To further increase their usability and distribution, some R/Bioconductor tools have been integrated into Galaxy (<https://galaxyproject.org/>), an open source, web-based platform for performing, reproducing, and sharing data analyses, which utilize a variety of bioinformatics tools^{4,5}. R/Bioconductor tool integration into Galaxy is a multi-step process that, although straightforward, poses unique challenges for both novice and advanced tool developers. The first major challenge is identifying and installing all of the dependencies needed for an R/Bioconductor tool. Each dependency needs to be available to Galaxy through its dependency management system, called the Tool Shed (<https://toolshed.g2.bx.psu.edu/>). If a dependency is not available in the Tool Shed, then it must be installed manually, which can often be difficult. The second major challenge is generating the required files/code for tool integration, which is a time-consuming process. Ensuring that the correct files are generated with correct syntax can be a frustrating task and is often the biggest hurdle for less-experienced developers. Given these issues, tool integration remains a daunting task for some developers, especially for those less familiar with command-line processes. Ideally, the R/Bioconductor tool integration process would be easy and intuitive for both novice and advanced developers.

Recognizing the need to improve the R/Bioconductor tool integration process, we outline here a guide and best practices (Table 1) for R/Bioconductor tool integration into Galaxy. Furthermore, we highlight the addition of new functionality to Planemo (<http://planemo.readthedocs.io/>)⁶, a suite of command-line utilities that assists in building and publishing Galaxy tools. To simplify generation of required tool integration files, we introduce a new Planemo command that creates template files based on an R/Bioconductor tool developed by the user. This new command also addresses dependency management issues by leveraging Bioconda (<https://bioconda.github.io>), a channel of the Conda package manager, which provides bioinformatics software⁷. Bioconda locates specific versioned R/Bioconductor packages via BioArchive (<https://bioarchive.galaxyproject.org/>)⁸ and CRAN (<https://cran.r-project.org/>)⁹ and includes these as requirements for using the R/Bioconductor tool. These new Planemo functionalities simplify R/Bioconductor tool integration and will enable developers to focus more attention on R/Bioconductor tool development and less on tool integration into Galaxy.

Methods

Here, we present a complete guide for developers to integrate an R/Bioconductor tool into Galaxy. We first describe the necessary components for wrapping an R/Bioconductor tool in Galaxy. Next, we outline the steps required for integrating the tool components into Galaxy, testing the tool, and executing the tool. Finally, we describe a simplified integration process using the new Planemo command, `bioc_tool_init`. The instructions provided in this guide assume that the developer has system access to Galaxy source code files (e.g. using a local instance or cloud-based instance of Galaxy), has an active Internet connection, and has installed the Planemo python package (v0.35.0 or later; <https://pypi.python.org/pypi/planemo/>)⁶. Furthermore, the user must have installed any packages that are dependencies for the R/Bioconductor tool being integrated.

Tool components and structure

An R/Bioconductor Galaxy tool is defined by four major components. The first component is a *Tool definition file* (Tool wrapper) in XML format, which provides the interface between Galaxy and the R/Bioconductor tool being integrated. The second component is a *Custom R file*, which calls the R/Bioconductor tool(s) to perform a particular analysis. The third component is a *Tool dependency file*, which tells Galaxy where to find the required tool dependencies. The fourth component is the *Test data* directory, which includes both input and output files that will be used to test the Custom R file. These four components should be organized using the following directory structure:

```
example_tool/
├── my_tool.R # Custom R file
├── my_tool.xml # Tool definition file
├── tool_dependencies.xml # Tool dependency file
├── test_data/ # Test data directory
│   ├── test_input.fq # Example fastq input file
│   ├── test_output.txt # Example output text file
│   └── ... # Additional input or output files
```

Tool definition file. The Tool definition file informs Galaxy how to handle parameters in the Custom R file. The value given to “name” in the file header appears in the Galaxy tool panel and should be set to a meaningful short description of what the tool does. The minimal structure of the Tool definition file contains seven key sections. The *requirements* section defines the tool dependencies needed to run the R script, and includes the version of R used to develop the tool. The *command* section defines the R command that is executed in Galaxy via the R interpreter. Importantly, input and output parameters are denoted as `$input1`, `$input2`, ... and `$output1`, `$output2`, ..., respectively, and the full path to the Custom R file is essential. The *inputs* section establishes how input parameters given to the Custom R file appear in Galaxy, while the *outputs* section defines the name and format of output files generated by the Custom R file. Each input and output parameter requires its own entry in the Tool definition file, and the values assigned to “name” should match those in the command section. The *tests* section defines the input parameters needed to test the R command and what output to expect as a result. This section is important for tool testing and

Table 1. Description of best practices for integrating R/Bioconductor tools into Galaxy. Described in this table are eight key “best practices” for developing R/Bioconductor tools that will be integrated into Galaxy.

Utilize descriptive sections in Tool definition file
Specifically, utilize the <i>test</i> section to identify which input parameters to use to test the R command and what output to expect as a result. Also, utilize the <i>help</i> section to provide text to describe the R/Bioconductor tool and any other useful information for the user.
Use CDATA to represent character data
In markup languages, such as XML, a CDATA section represents content that should be interpreted purely as character data. Using CDATA is highly recommended, for example, for text in the <i>help</i> and <i>command</i> sections of the Tool definition file.
Choose output formats recognized by Galaxy
Output generated by the Custom R file should be saved in a format that is present in the list of Galaxy-recognized datatypes. By using one of these datatypes, the user can view the file in Galaxy and perform manipulations on the data using other Galaxy tools. The list of available Galaxy datatypes can be seen in the Admin panel under “Datatypes Registry”. Galaxy now also saves files as Rdata files, and the feature to visualize these files is under development.
Pass command-line arguments with flags
In the Custom R file examples presented here, the R package <i>getopt</i> is used to add command-line arguments with flags. It is good practice to pass arguments using flags - this is currently required by <code>bioc_tool_init</code> - because parameters are explicitly defined and not inferred.
Implement error handling
Handling error messages upon execution of an R script is particularly important. If an error message or any message generated by an R script is printed to stdout instead of stderr, Galaxy interprets this as the tool correctly producing an output file and does not know whether or not the tool failed. By specifically telling R to output error messages to stderr, Galaxy is able to distinguish between a failed job and a successful job. Include the following lines at the beginning of a Custom R file to send R errors to stderr and to handle UTF8 errors:
<pre>## Redirect R error handling to stderr. options(show.error.messages=F, \ error=function() {cat(geterrmessage(), \ file=stderr());q("no",1,F)}) ## Avoid crashing Galaxy with a UTF8 error on German LC settings. loc <- Sys.setlocale("LC_MESSAGES", "en_US.UTF-8")</pre>
Avoid flooding stdout
It is recommended to suppress messages to stdout when loading R/Bioconductor packages. To do this, include the following lines in the Custom R file:
<pre>## Load required libraries without seeing messages suppressPackageStartupMessages({ library("getopt") library("affy") })</pre>
Allow verbose output for debugging
Toggle verbose outputs via the Custom R file and <i>getopt</i> package to allow for easier debugging. Intermittent status messages can also be written while the R script is processing large datasets. This allows the Custom R file to be usable outside of Galaxy as well. The example below includes a statement to print the defined parameters, which will be viewable in the history panel upon execution of tool.
<pre>## Invoke by executing "Rscript my_r_tool_again.R --verbose TRUE" option_specifications <- matrix(\ c("verbose", "v", 2, "logical"), \ byrow=TRUE, ncol=4) options <- getopt(option_specifications) ## Toggle verbose option if (options\$verbose) { cat ("Print something useful to stdout to show \ how script is running in Galaxy.\n") }</pre>

Avoid writing R code in <configfiles>

It is recommended to keep the Custom R file and the Tool definition file separate and avoid placing R code in a configfiles tag within the Tool definition file. If the R code is embedded inside the Tool definition file, it cannot be used outside of Galaxy due to syntax differences and it cannot be easily tested. This following example should be avoided:

```
# In Tool definition file, after the <command> tag
<configfiles>
  <configfile name="Extract expression"><![CDATA[
    # Read in data
    inputfile <- as.character(options$input)
    data <- ReadAffy(fileNames = inputfile)
    # More R code...
  ]]>
</configfile>
</configfiles>
```

debugging. The *help* section should be used to describe the R/Bioconductor tool and will appear at the bottom of the Galaxy tool form. Finally, appropriate references for the tool can be provided using the *citations* section. References can be cited, for example, using a DOI or a BibTeX entry. An example Tool definition file for an R/Bioconductor tool that enumerates *k*-mers in a fastq file is available as [Supplementary File 1](#). This tool will subsequently be referred to as “Kmer_enumerate” and will be referenced throughout the remaining sections of this guide.

Custom R file. The Custom R file establishes the R environment and informs Galaxy what R command(s) to execute. The first section of this file contains a header of information that handles error messages, loads required R libraries, and parses options. These requirements are needed for every R/Bioconductor tool being integrated; however, the list of imported R libraries will be specific to each tool. The next section defines the list of parameters to pass to the R command, including input and output parameters. Each parameter, if using the getopt command line parsing library, requires a unique name, a unique single letter designation, a flag indicating whether the parameter is required (0=no argument; 1=required; 2=optional), and the parameter type (e.g. character, integer, float). Optionally, variable names and values can be printed to standard output (stdout), which can be viewed in Galaxy when the tool executes. While not required, these printed statements can assist in debugging and inform whether the R/Bioconductor tool was executed correctly. The final section contains the R command(s) needed to execute the R/Bioconductor tool. A Custom R file for the Kmer_enumerate tool is available as [Supplementary File 2](#). This tool uses the R/Bioconductor package seqTools¹⁰ to read in a fastq file of DNA sequences, count the number of *k*-mers in the sequences where the value *k* is supplied by the user, and output the *k*-mers and their counts.

Unlike running standalone R scripts in the command line or using a graphical interface, it is not necessary to define the working directory (e.g. using `setwd()`) in the Custom R file. By default, Galaxy executes the R script in the same directory where the

files are located. Similarly, Galaxy writes output files to the same directory, which enables the results to be displayed in the Galaxy history panel. Before attempting to integrate an R/Bioconductor tool, it is strongly recommended to test the Custom R file as a standalone script. For example, the Kmer_enumerate Custom R file can be executed in the command line using the following (test input and output files are available as [Supplemental File 3](#) and [Supplemental File 4](#), respectively):

```
cd /path/to/example_seqTools/
Rscript my_seqTools_tool.R \
  --input1 test_data/test_input.fq \
  --input2 2 \
  --output test_data/test_output.txt
```

Tool dependency file. The Tool dependency XML file informs Galaxy where to find the required tool dependencies and should explicitly reference each of the requirements listed in the Tool definition file. It is important to differentiate between the requirements tag in the Tool definition file and the Tool dependency file: the Tool definition file identifies *what* dependencies are needed, and the Tool dependency file identifies *where* to get the dependencies. For example, many of the required tools - R/Bioconductor or otherwise - are available in the Galaxy Tool Shed. In the Tool dependency file, each requirement is listed under its own repository tag with its “name” and “owner” parameters as they appear in the Tool Shed. Available Tool Shed tools can be viewed at <https://toolshed.g2.bx.psu.edu/>. A Tool dependency file for Kmer_enumerate is supplied as [Supplementary File 5](#).

Test data. The Test data directory includes data file(s) intended as input to test the R script and any expected output data file(s). During testing, Galaxy runs the R/Bioconductor tool with the input files in the test data directory and compares the output with the output file(s) in the same directory to ensure that the tool is producing expected results. As an added benefit, including testing data for the tool provides an example for other users of the data formats needed to run the tool.

For tools that output plots, generating test data becomes an issue. To test whether R figures and plots are generated correctly, they should be saved as PNG files instead of PDF. Saving plots and figures as PDF files is a common practice in R/Bioconductor packages, but when PDF files are generated, they are time-stamped. Galaxy will not consider two PDFs identical - even if they display the same image - if they were generated at different times and thus have different timestamps.

Tool integration

The following steps outline how to integrate the new R/Bioconductor tool into Galaxy after the tool files have been generated:

Step 1: Assemble the Tool definition file, Custom R file, Tool dependency file, and Test data directory with test data files in a single directory. Update the Tool definition file to provide the full path where appropriate. Alternatively, if the tool directory is saved in the `$GALAXY_ROOT/tools/` directory, a relative path is sufficient.

Step 2: Copy the Tool configuration file `tool_conf.xml.sample`, if it does not already exist, and save it as `tool_conf.xml`.

```
cp $GALAXY_ROOT/config/tool_conf.xml.sample \
  $GALAXY_ROOT/config/tool_conf.xml
```

Step 3: Modify `tool_conf.xml` by adding a new section under which the integrated tool will exist. The value given to “name” in the Tool configuration file will appear in the tool panel, and the value given to “name” in the Tool definition file will appear under this new section. Provide the full path to the Tool definition file if the tool directory is not in `$GALAXY_ROOT/tools/`. Otherwise, the relative path is sufficient.

```
<section id="testingtools" name="Testing Tools">
  <!-- Full path: -->
  <tool file="/path/to/example_seqTools/my_seqTools_tool.xml" />
  <!-- Relative path: -->
  <tool file="example_seqTools/my_seqTools_tool.xml" />
</section>
```

Step 4: Restart Galaxy to integrate the modified `tool_conf.xml` file.

Tool testing and execution

Including test cases for newly integrated tools - while not strictly required - is highly recommended because it enables easier debugging and ensures that the tool is working as expected. To test, for example, the `Kmer_enumerate` tool, first upload the testing input file (`test_input.fq.gz`) to Galaxy. Choose the `Kmer_enumerate` tool from the tool panel, update the input file and *k*-mer parameters, and execute the tool. In this example, when `Kmer_enumerate` is executed, `test_input.fq.gz` is passed by the `--input1` argument and the *k*-mer value is passed by the `--input2` argument in the Tool definition file to the Custom R file. The Custom R file executes and sends the results

back to the Tool definition file. The output of the tool is then available for viewing in the Galaxy history panel. Since Galaxy interprets any output written to standard error (stderr) as a failed job, it is important that developers ensure that any output (e.g. files, messages) generated by the R script is sent to stdout and not stderr.

When the developer is satisfied that the integrated tool executes correctly, the tool is ready to be used for data analysis. At this point, the newly integrated R/Bioconductor tool can be published in the Galaxy Tool Shed, so that it will be available to the Galaxy community. Detailed instructions for how to submit tools for publishing in the Galaxy Tool Shed can be found online at <http://planemo.readthedocs.io/en/latest/publishing.html>.

Use cases

The above guide outlines a straightforward approach for integrating a relatively simple R/Bioconductor tool into Galaxy. However, generating the Tool definition XML file and ensuring all required dependencies, which are available to Galaxy, remain a difficult task. To simplify R/Bioconductor tool integration and address these remaining challenges, we have added a new command to Planemo v0.34.1, a suite of command-line utilities to assist in building and publishing Galaxy tools. We also take advantage of the package manager Conda v4.2.4 and, specifically, the Bioconda channel¹¹ of Conda, which distributes bioinformatics-related software. The new Planemo command, `bioc_tool_init`, creates a Bioconda recipe of all dependencies for the given R/Bioconductor tool and writes the path to this recipe to the Tool definition file. This command eliminates the need for recursively parsing the R/Bioconductor dependency tree to create the Tool dependency file. Further, this approach has the added benefit of creating an artifact, which, while created to use with a Galaxy tool, is potentially useful outside of Galaxy by anyone using Bioconda. Additional arguments to the `bioc_tool_init` command specifically address R/Bioconductor tool integration and are described in the two use cases below.

The `bioc_tool_init` command functions by invoking another new Planemo command, `bioc_conda_recipe_init`. In this command, Bioconda uses BioArchive v2.35.0, a Bioconductor package version archive, to retrieve the correct package versions if they exist. The `bioconductor_skeleton.py` script has been modified to not only find missing R/Bioconductor package dependencies, but also create them in the local Bioconda repository specified by the user. BioArchive improves reproducibility of the Bioconda recipes of different versions of the same Bioconductor package.

Here, we describe how to generate a Tool definition file using `bioc_tool_init` for a Custom R script that uses the R/Bioconductor package `affy` v1.52.0¹². We assume that the user has installed Planemo (v0.35.0 or newer) along with the requirements for the software and has git configured with the appropriate ssh keys (<https://help.github.com/articles/generating-an-ssh-key/>). We first describe the simplest integration scheme (Case 1) requiring only one parameter, followed by more complex integration schemes (Case 2) with multiple parameters.

Case 1: Generating a Tool definition file and Bioconda recipe with a single parameter

The following command is the simplest way to generate a Tool definition file and Bioconda recipe for integrating a Custom R file in Galaxy (Supplementary File 6–Supplementary File 9). This tool, subsequently referred to as “Extract_expression”, implements the affy package to extract probe expression levels from an Agilent CEL file generated by a microarray experiment.

```
planemo bioc_tool_init \
  --command "Rscript /path/to/my_affy_tool.R \
    --input test_input.CEL \
    --output test_output.txt"
```

Only the `--command` option is required here because all of the necessary information for building and running the tool is present in the Rscript call. The full path to the Custom R file should be given so that the Tool definition file can correctly locate it. In addition, extensions for all `--input` and `--output` files are also required, as they are needed to populate the format parameter in the inputs and outputs tags in the Tool definition file. The key to this usage of the `bioc_tool_init` command is that the R command given to `--command` successfully executes in the command line. The example input¹³ and output files used here are available as Supplementary File 10 and Supplementary File 11, respectively. In a similarly simple example, each parameter in the Rscript command can be given to `bioc_tool_init` using the following arguments:

```
planemo bioc_tool_init \
  --rscript /path/to/my_affy_tool.R \
  --input test_input.CEL \
  --output test_output.txt
```

In both examples, the `bioc_tool_init` command first clones the Bioconda repository in the home directory and creates a new recipe for each dependency in `$HOME/bioconda-recipes/recipes/` if it does not exist. This allows the user to have a local copy of the recipe and all of the Bioconda package dependencies for that recipe. The `bioc_tool_init` command then creates a new Tool definition file, `my_affy_tool.xml`, in the current directory with the newly generated Bioconda recipe as a requirement. The `bioc_tool_init` command is used to create, not update, the dependency requirements and provide a suitable blueprint for the Tool definition file. Semi-automated creation of the Tool definition XML file enables users to quickly generate usable code for R/Bioconductor tool integration.

Case 2: Generating a Tool definition file and Bioconda recipe with a multiple parameters

Additional options are available for `bioc_tool_init` and are strongly recommended for generating a tool that follows Galaxy tool development best practices. For example, the `--name` option sets the name that will appear in the Galaxy tool panel (defaults to the name of the Tool definition file) and should be a brief statement of what the tool does. The `--description` option provides additional information about the tool's function and appears immediately after the tool name in the tool panel.

The `--help_text` option populates a field in the Galaxy tool form that provides additional information about what the tool does. Finally, developers should utilize the `--doi` option to include citation information for the tool in the tool form. An exhaustive list of available arguments can be found by using `planemo bioc_tool_init --help`. The following is an example command to generate the Tool definition file and Bioconda recipe for the Extract_expression tool:

```
planemo bioc_tool_init \
  --command "Rscript /path/to/my_affy_tool.R \
    --input test_input.CEL \
    --output test_output.txt" \
  --name "Extract Expression" \
  --description "values from CEL Agilent microarray data" \
  --help_text "This tool reads in Agilent CEL data and \
    outputs probe expression values." \
  --doi "10.1093/bioinformatics/btg405" \
  --tool my_affy_tool_Case2.xml
```

Using this command, the tool dependencies and requirements in `my_affy_tool.R` are automatically written to the Tool definition file (Supplementary File 12). As in Case 1, appropriate input and output data formats are inferred from the `--input` and `--output` arguments given to `--command`. It is important to note that the Tool definition file and the Bioconda recipe generated by the `planemo bioc_tool_init` command are meant to be working, usable code for tool integration. However, the files are not 100% complete in terms of following best practices for Galaxy tool development, and may require additional work to reach the standards for which Galaxy tools are published to the Tool Shed. Galaxy R/Bioconductor tool developers are strongly encouraged to meet best practice standards for any tool.

Discussion

Integrating R/Bioconductor tools into Galaxy can be challenging for both novice and advanced tool developers, but it is an important part of increasing the availability and reproducibility of research tools for the scientific community. We provide here a complete guide for R/Bioconductor tool integration that includes: (1) a description of the components needed to integrate the tool, (2) step-by-step instructions for incorporating the tool components into Galaxy, (3) examples of how to use the new `bioc_tool_init` Planemo command for easier tool integration, and (4) best practices for R/Bioconductor tool integration. A more detailed guide for R/Bioconductor tool integration into Galaxy is available on GitHub at <https://github.com/nturaga/bioc-galaxy-integration/blob/master/README.md>. By providing a way to semi-automate the integration process, we hope that R/Bioconductor tool developers can focus more on developing new and essential tools rather than on how to integrate them into Galaxy.

A key feature of the simplified tool integration method described in this work is the addition of the `bioc_tool_init` command to Planemo. This new capability specifically improves tool

integration for developers in two ways. First, the `bioc_tool_init` command generates nearly complete Tool dependency files and Bioconda recipes by directly parsing the Custom R script being integrated, eliminating the need for developers to manually update the correct tool names and versions in all the tool files. Second, the `bioc_tool_init` command alleviates dependency management issues by recursively identifying and installing all required tool dependencies using Bioconda. This ensures that tool dependencies are compatible with and accessible across different platforms and eliminates the need for developers to manually install all required dependencies. We hope that these improvements will encourage more R/Bioconductor tool developers to share and publish their tools on Galaxy.

Future work for improving ease of R/Bioconductor tool integration into Galaxy includes improving the `bioc_tool_init` command to automate more tasks. For example, we are currently working on a functionality that automatically generates example test cases to include in the Tool definition XML file. We are also working on extending the `bioc_tool_init` command to handle integration of multiple R/Bioconductor functions by passing a formatted text file to `bioc_tool_init`. We are also working on a Planemo command that automatically submits a wrapped R/Bioconductor tool to the public Tool Shed. Finally, development of a Planemo command that can automatically wrap an entire R/Bioconductor package based on a published vignette would be ideal for quickly integrating and publishing Galaxy-wrapped tools. These and other improvement are currently undergoing development.

Data and software availability

Automated build available from: <https://hub.docker.com/r/nitesh1989/bioc-galaxy-integration/>
Latest source code: <https://github.com/nturaga/bioc-galaxy-integration>

Archived source code as at the time of publication: DOI, [10.5281/zenodo.166551](https://doi.org/10.5281/zenodo.166551)¹⁴

License: Academic Free License version 3.0

More information on tool building in Galaxy and additional best practices are available at <http://planemo.readthedocs.io/en/latest/writing.html>. Planemo documentation can be found at <https://github.com/galaxyproject/planemo>.

Author contributions

NT and MAF designed and implemented the features presented with advice from DB, JC, AN, and JT. JC implemented the Planemo tool on which this work is built. NT, MAF, and DB wrote the paper. Members of the Galaxy Team developed the Galaxy framework on which this work relies and provided advice on the project. All authors have agreed to the final content.

Competing interests

No competing interests were disclosed.

Grant information

This project is funded in part by the National Institutes of Health [U41 HG006620].

The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Acknowledgments

We are grateful to the Galaxy Team (<https://wiki.galaxyproject.org/GalaxyTeam>) and the developer community for the Galaxy Framework for their guidance on this project. We would also like to thank the developers of Bioconda for providing a great service to the community. We give special mention to Ryan Dale for developing the script to create Bioconda recipes for Bioconductor packages. We also thank the contributors of BioRchive.

Supplementary files

Supplementary File 1: Tool definition file for Kmer_enumerate. An example Tool definition file for the Kmer_enumerate tool, which enumerates *k*-mers in a fastq file. This file can be opened with any text editor.

[Click here to access the data.](#)

Supplementary File 2: Custom R file for Kmer_enumerate. An example Custom R file for the Kmer_enumerate tool. This file can be opened with any text editor.

[Click here to access the data.](#)

Supplementary File 3: Input for Kmer_enumerate. An example input fastq file for the Kmer_enumerate tool. This fastq formatted file has been compressed with gzip. The file must be uncompressed (using gunzip) before it can be opened with any text editor.

[Click here to access the data.](#)

Supplementary File 4: Output for Kmer_enumerate. An example output text file for the Kmer_enumerate tool. This file can be opened with any text editor.

[Click here to access the data.](#)

Supplementary File 5: Tool dependencies for Kmer_enumerate. An example Tool dependencies file for the Kmer_enumerate tool. This file can be opened with any text editor.

[Click here to access the data.](#)

Supplementary File 6: Tool definition file for Extract_expression Case 1. An example Tool definition file for the Extract_expression tool created using Case 1. This file can be opened with any text editor.

[Click here to access the data.](#)

Supplementary File 7: Yaml file for Extract_expression. An example Bioconda recipe yaml file for the Extract_expression tool created using Case 1. This file can be opened with any text editor.

[Click here to access the data.](#)

Supplementary File 8: Bash file for Extract_expression. An example Bioconda recipe bash file for the Extract_expression tool created using Case 1. This file can be opened with any text editor.

[Click here to access the data.](#)

Supplementary File 9: Custom R file for Extract_expression. An example Custom R file for the Extract_expression tool created using Case 1. This file can be opened with any text editor.

[Click here to access the data.](#)

Supplementary File 10: Input CEL file for Extract_expression. An example input CEL file for the Extract_expression tool¹³. This file is a binary data file created by Affymetrix DNA microarray image analysis software, so it cannot be viewed with a text editor. The contents of this CEL file can be accessed in R via the readAffy() function of the affy Bioconductor package¹².

[Click here to access the data.](#)

Supplementary File 11: Output for Extract_expression. An example output text file for the Extract_expression tool. This file can be opened with any text editor.

[Click here to access the data.](#)

Supplementary File 12: Tool definition file Extract_expression Case 2. An example Tool definition file for the Extract_expression tool created using Case 2. This file can be opened with any text editor.

[Click here to access the data.](#)

References

- Huber W, Carey VJ, Gentleman R, *et al.*: **Orchestrating high-throughput genomic analysis with Bioconductor.** *Nat Methods.* 2015; 12(2): 115–121.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
- Gentleman RC, Carey VJ, Bates DM, *et al.*: **Bioconductor: open software development for computational biology and bioinformatics.** *Genome Biol.* 2004; 5(10): R80.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
- R Core Team: **R: A language and environment for statistical computing.** R Foundation for Statistical Computing, Vienna, Austria. 2015.
[Reference Source](#)
- Giardine B, Riemer C, Hardison RC, *et al.*: **Galaxy: a platform for interactive large-scale genome analysis.** *Genome Res.* 2005; 15(10): 1451–1455.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
- Goecks J, Nekrutenko A, Taylor J, *et al.*: **Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences.** *Genome Biol.* 2010; 11(8): R86.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
- Planemo. GitHub. Cited 12 Oct 2016.
[Reference Source](#)
- Conda. GitHub. Cited 12 Oct 2016.
[Reference Source](#)
- Turaga N: **bioArchive: enabling reproducibility of Bioconductor based analyses [v1; not peer reviewed].** *F1000Res.* 2015; 4(ISCB Comm J): 370(poster).
[Publisher Full Text](#)
- CRAN: **The Comprehensive R Archive Network.** Cited 12 Oct 2016.
[Reference Source](#)
- Kaisers W: **seqTools: Analysis of nucleotide, sequence and quality content on fastq files.** R package version 1.6.0. 2013.
[Reference Source](#)
- Bioconda. GitHub. Cited 12 Oct 2016.
[Reference Source](#)
- Gautier L, Cope L, Bolstad BM, *et al.*: **affy--analysis of Affymetrix GeneChip data at the probe level.** *Bioinformatics.* 2004; 20(3): 307–315.
[PubMed Abstract](#) | [Publisher Full Text](#)
- Havis E, Bonnin MA, Olivera-Martinez I, *et al.*: **Transcriptomic analysis of mouse limb tendon cells during development.** *Development.* 2014; 141(19): 3683–3696.
[PubMed Abstract](#) | [Publisher Full Text](#)
- Chilton J, Cock P, Rasche E, *et al.*: **Galaxy Planemo 0.35.0 [Data set].** *Zenodo.* 2016.
[Data Source](#)

Open Peer Review

Current Peer Review Status:

Version 1

Reviewer Report 28 December 2016

<https://doi.org/10.5256/f1000research.10589.r18534>

© 2016 Noushmehr H et al. This is an open access peer review report distributed under the terms of the [Creative Commons Attribution License](#), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Houtan Noushmehr

¹ Department of Neurosurgery, Henry Ford Hospital, Detroit, MI, USA

² OMICS Laboratory, Department of Genetics, Ribeirão Preto Medical School, University of São Paulo, São Paulo, Brazil

Tiago Silva

OMICS Laboratory, Department of Genetics, Ribeirão Preto Medical School, University of São Paulo, São Paulo, Brazil

The manuscript by Turaga *et al.* is a useful guide for novice and advanced users of R/Bioconductor and Galaxy to incorporate any R/Bioconductor packages within the popular Galaxy software. We believe that by allowing users to integrate any Bioconductor package within Galaxy will add enormous utility and advancement for any analyst. We envision users with little experience with R/Bioconductor would eventually be able to seek support to integrate any new Bioconductor package and thus incorporate the workflow within their data analysis pipeline within Galaxy. The authors have done an excellent job by highlighting several key best practices to achieve a reliable integration as well as the necessary structure to integrate a new BioC packages with example files highlighted within their supplemental data. This integration is explained through two distinct processes; A manual version and a semi-automated process that utilizes the tool Planemo (<https://github.com/galaxyproject/planemo>), a command-line suite of tool to assist in developing tools for the Galaxy Project. The authors envision a more streamlined version of their tool with subsequent improvements, and the expectations will eventually lead to a larger base of users (little and no experience in R to advanced users).

The text is well-written and is well structured, and we were able to follow the manual integration. However, due in part to a non-working version of the latest build from the GitHub repository, we were unable to implement the planemo tool `bioc\_tool\_init` and thus we can not provide a thorough evaluation of the tool. We expect that once the tool is available, we can provide a proper evaluation by integrating a random BioC package.

Some minor points:

1. In the tool dependency file description, we were unable to understand how and where one

would obtain the name and owner of each dependency. For example, we were unable to find the name and owner of seqTools in <https://toolshed.g2.bx.psu.edu/>.

2. In the tool integration section, the code has the full path and the relative path for the file in the section. We feel this is redundant. Please consider modifying the code.
<https://gist.github.com/tiagochst/a7b0ff56a864ca1ae2d5eaeae82db9b>
3. This issue is probably not the author's fault, but the supplementary files do not possess the same name as the example. For this reason, the user has to download and rename them to execute the guide. Maybe adding a compressed zip with the correct names would help.

Major problems:

Planemo: `bioc\_tool\_init` unavailable

As explained in the methods section we were able to install the latest version of Planemo (0.36.1). However, the command `bioc\_tool\_init` was unavailable and thus we were unable to evaluate the command by testing it with our packages. We followed the following instructions posted here: <https://github.com/galaxyproject/planemo> (Here is a screenshot of the version and the problem we experienced during install <https://goo.gl/kyh33j>). If further steps are required to install the `bioc\_tool\_init` function, we feel it should be well documented either in git or this manuscript. Due to its unavailability to test and confirm its practicality, we are unable to accept this MS in its current form for publication. Once the tool is available, we can provide a complete evaluation.

Competing Interests: No competing interests were disclosed.

We confirm that we have read this submission and believe that we have an appropriate level of expertise to confirm that it is of an acceptable scientific standard, however we have significant reservations, as outlined above.

Author Response 09 Jan 2017

Mallory Freeberg, Johns Hopkins University, Baltimore, USA

Thank you for your insightful and useful feedback!

We are in the process of addressing the minor concerns and issues raised here and will be releasing a version 2 of the manuscript shortly. We look forward to hearing about your experience wrapping an R/BioC tool in Galaxy!

In the meantime, we have addressed your major concern of the `bioc_tool_init` command not being installed with Planemo. You should be able to see this command now by installing Planemo via pip or GitHub.

Again, thank you for your feedback!

Best,

Mallory

Competing Interests: No competing interests were disclosed.

Reviewer Report 21 December 2016

<https://doi.org/10.5256/f1000research.10589.r18535>

© 2016 Stewart P et al. This is an open access peer review report distributed under the terms of the [Creative Commons Attribution License](#), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Paul A. Stewart

Department of Thoracic Oncology, H. Lee Moffitt Cancer Center & Research Institute, Tampa, FL, USA

Dekai Rohlsen

Department of Thoracic Oncology, H. Lee Moffitt Cancer Center & Research Institute, Tampa, FL, USA

This manuscript, as the title implies, presents a guide and best practices for Bioconductor tool integration into the Galaxy web environment. Some Bioconductor tools have been integrated into Galaxy, but there are still a number of tools that are not supported and need to be installed by a Galaxy administrator in a multi-step process. The two biggest challenges for any type of tool integration are introduced: installation of all tool dependencies and correctly generating tool component files. The authors provide a table of best practices to help address these challenges, but more importantly the authors introduce Planemo, a wonderful command-line utility suite, which greatly aids in the creation and publishing of Galaxy tools.

The **Methods** section does an excellent job at providing a guide for integrating Bioconductor tools, and the **Use cases** section, together with sample commands and provided supplementary files, shows the reader the benefits and ease of use of the Planemo suite. We have tested the provided sample files and code, and it behaves as described in our hands.

The manuscript is very well written and will serve as a great reference for seasoned developers wishing to integrate R/Bioconductor tools into Galaxy, and the clearly written explanations as well as use cases will help ease newcomers into working with Galaxy. All examples provided are with R/Bioconductor, but this content could easily be used as a manual integrating other tools or command line scripts outside of Bioconductor. This is a great development for Galaxy documentation, and we are happy to recommend the article for approval.

Competing Interests: No competing interests were disclosed.

We confirm that we have read this submission and believe that we have an appropriate level of expertise to confirm that it is of an acceptable scientific standard.

Author Response 09 Jan 2017

Mallory Freeberg, Johns Hopkins University, Baltimore, USA

Thank you for your insightful and useful feedback!

We are thrilled that our use cases and sample files worked for you. We are planning some improvements to the manuscript and will release a version 2 shortly.

Cheers,
Mallory

Competing Interests: No competing interests were disclosed.

The benefits of publishing with F1000Research:

- Your article is published within days, with no editorial bias
- You can publish traditional articles, null/negative results, case reports, data notes and more
- The peer review process is transparent and collaborative
- Your article is indexed in PubMed after passing peer review
- Dedicated customer support at every stage

For pre-submission enquiries, contact research@f1000.com

F1000Research