BioInstaller: a comprehensive R package to construct interactive and reproducible biological data analysis applications based on the R platform

Jianfeng Li 1, Bowen Cui 1, Yuting Dai 1,2, Ling Bai 1, Jinyan Huang Correspond.

1 State Key Laboratory of Medical Genomics, Shanghai Institute of Hematology, National Research Center for Translational Medicine, Rui-jin Hospital, Shanghai Jiao Tong University School of Medicine, Shanghai Jiao Tong University, Shanghai, China
2 School of Life Sciences and Biotechnology, Shanghai Jiao Tong University, Shanghai, China

Corresponding Author: Jinyan Huang
Email address: jinyan@shsmu.edu.cn

The increase in bioinformatics resources such as tools/scripts and databases poses a great challenge for users seeking to construct interactive and reproducible biological data analysis applications. Here, we propose an open-source, comprehensive, flexible R package named BioInstaller that consists of the R functions, Shiny application, the HTTP representational state transfer (REST) application programming interfaces (APIs), and a docker image. BioInstaller can be used to collect, manage and share various types of bioinformatics resources and perform interactive and reproducible data analyses based on the extendible Shiny application with Tom's Obvious, Minimal Language (TOML) and SQLite format databases. The source code of BioInstaller is freely available at our lab website, http://bioinfo.rjh.com.cn/labs/jhuang/tools/bioinstaller, the popular package host GitHub, https://github.com/JhuangLab/BioInstaller, and the Comprehensive R Archive Network (CRAN), https://CRAN.R-project.org/package=BioInstaller. In addition, a docker image can be downloaded from DockerHub (https://hub.docker.com/r/bioinstaller/bioinstaller).
BioInstaller: a comprehensive R package to construct interactive and reproducible biological data analysis applications based on the R platform

Jianfeng Li\textsuperscript{1,2,*}, Bowen Cui\textsuperscript{1,*}, Yuting Dai\textsuperscript{2}, Ling Bai\textsuperscript{1}, and Jinyan Huang\textsuperscript{1}

\textsuperscript{1}State Key Laboratory of Medical Genomics, Shanghai Institute of Hematology, National Research Center for Translational Medicine, Rui-Jin Hospital, Shanghai Jiao Tong University School of Medicine, Shanghai Jiao Tong University, Shanghai, China

\textsuperscript{2}School of Life Sciences and Biotechnology, Shanghai Jiao Tong University, Shanghai, China

* These authors contributed equally to this work.

Corresponding author: Jinyan Huang, jinyan@shsmu.edu.cn

ABSTRACT

The increase in bioinformatics resources such as tools/scripts and databases poses a great challenge for users seeking to construct interactive and reproducible biological data analysis applications. Here, we propose an open-source, comprehensive, flexible R package named BioInstaller that consists of the R functions, Shiny application, the HTTP representational state transfer (REST) application programming interfaces (APIs), and a docker image. BioInstaller can be used to collect, manage and share various types of bioinformatics resources and perform interactive and reproducible data analyses based on the extendible Shiny application with Tom's Obvious, Minimal Language (TOML) and SQLite format databases. The source code of BioInstaller is freely available at our lab website, http://bioinfo.rjh.com.cn/labs/jhuang/tools/bioinstaller, the popular package host GitHub, https://github.com/JhuangLab/BioInstaller, and the Comprehensive R Archive Network (CRAN), https://CRAN.R-project.org/package=BioInstaller. In addition, a docker image can be downloaded from DockerHub (https://hub.docker.com/r/bioinstaller/bioinstaller).

INTRODUCTION

With the rapid development of new bioscience technology, particularly next-generation sequencing (NGS), volumes of “omics” data have been generated, such as the 1000 Genomes Project, The Cancer Genome Atlas (TCGA), and Genotype-Tissue Expression (GTEX) (Abecasis et al. 2012; Cancer Genome Atlas Research et al. 2013; Consortium 2013; Sanchez-Vega et al.)
The bioinformatics tools and databases required for the downstream data analysis are also increasing at a phenomenal rate. R language, as the most popular programming language for statistics, biological data analysis, and big data, has enabled diverse and free R packages (>14000) for different types of applications, such as high throughput sequencing data analysis (e.g. Bioconductor) (Gentleman et al. 2004) and the development of web applications (e.g. Shiny framework) (Chang et al. 2015). With the development of web technologies and the release of the R web developmental framework Shiny, the number of interfaces available to R users has increased. However, due to the lack of high-performance and open-source cloud platforms based on R (e.g., Galaxy for Python users) (Afgan et al. 2016), it is still difficult for R users, especially those without web development skills, to construct interactive and reproducible biological data analysis applications supporting the upload and management of files, long-time computation, task submission, tracking of output files, exception handling, logging, export of plots and tables, and extendible plugin systems.

Another common problem usually faced by R and other programming platform users (e.g., the team of Galaxy) (Afgan et al. 2016) is how to acquire and share certain bioinformatics resources quickly and accurately. Numerous bioinformatics tools (e.g., primer design, sequence alignment, variant calling and annotation) or scripts (e.g., data format conversion, text processing) are scattered around world web hosts. Biomedical databases are facing the same situation. For example, genome sequences (e.g., hg19/hg38 for human, mm9 and mm10 for mouse) are mainly deposited in the UCSC Genome Browser and National Center for Biotechnology Information (NCBI, https://www.ncbi.nlm.nih.gov/) (Tyner et al. 2017). The best-known gene and transcript annotation resources are provided by GENCODE and the RefSeq database (Derrien et al. 2012; O'Leary et al. 2016). Genetic variants annotation databases, mainly cancer and Mendelian disorder related, are hosted by the original projects, e.g. TCGA, and various down-stream tools, e.g., ANNOVAR, Variant Effect Predictor (VEP), Oncotator (McLaren et al. 2016; Ramos et al. 2015; Wang et al. 2010). Bioconductor is a popular bioinformatics R community for sharing genetic variants and other types of bioinformatics annotation databases via R package (Gentleman et al. 2004), but it is difficult for users to share many types of tools/scripts and databases if they do not have the capability of packing their own tools/scripts and databases. In most cases, these resources are isolated and can only be accessed via a command line tool such as rsync (https://rsync.samba.org/) or wget (http://www.gnu.org/software/wget/), to request the
corresponding Uniform Resource Locators (URLs). Software distribution tools that do not demand root privileges, such as conda (https://github.com/conda/conda) and spack (Gamblin et al. 2015), have greatly improved the acquisition of bioinformatics software. However, considering the huge growth of tools/scripts and databases required for bioinformatics data analysis, the resources supported by these software distribution tools are far from sufficient. Users also need more experience to use these different package management tools under command line environment.

Here, we present an open-source, comprehensive, flexible bioinformatics platform named BioInstaller that can be used to collect, manage and share various types of bioinformatics resources and to perform interactive and reproducible data analyses. By utilizing a simplified and standard TOML format configuration file with extra parse functions, the developers and users can freely and unreservedly share their public or internal bioinformatics tools/scripts and databases online on the GitHub repository or other hosts. In addition, users can easily obtain access to pooled bioinformatics resources via the diverse interfaces of BioInstaller, which includes R functions, the Shiny application (Chang et al. 2015) and HTTP representational state transfer (REST) application programming interfaces (APIs) that are rarely adopted in other similar tools. As a practical demonstration, we collected 157 tools/scripts and 110 databases specifically related to genetic variants annotation using the BioInstaller-defined configuration files. Notably, we developed a Shiny application to support functions including system monitoring, the logging system, file management, the queue system, and so on. This application can easily be reused in other Shiny applications. We expect the BioInstaller package and the practices in this work to reduce the difficulty of constructing the interactive and reproducible biological data analysis applications for R users, and to further improve the interactivity and reproducibility of bioinformatics data analysis.

MATERIAL AND METHODS

Design and development of BioInstaller

BioInstaller was designed as an interactive R package to collect, manage and share various types of bioinformatics resources and perform interactive and reproducible data analyses. BioInstaller contains the R functions and the Shiny application (Chang et al. 2015) and REST APIs (Figure 1). Both R and other programming platform users can utilize the functions of BioInstaller, such as by downloading bioinformatics tools/scripts and databases and performing statistical analysis and visualization. The R and Shiny interfaces of BioInstaller were mainly developed in R language.
and utilize the HTML/CSS and JavaScript languages. To run an instance of BioInstaller, the R program and extra dependent R packages are required. Travis CI (https://www.travis-ci.org/) was used to automatically test the R functions on Linux and MAC OSX platforms. Periodically, the tested and updated BioInstaller package is submitted to CRAN with an increased version number, e.g., from v3.3.3 to v3.3.4. Both the open and restricted bioinformatics resources can be integrated using the TOML format configuration file. The configuration files can also be used in other programming language platforms to access desired masteries by using a unique item name, such as ‘bwa’, ‘gatk’, ‘annovar’, ‘db_annovar_1000g’, ‘db_annovar_gtex’, etc. A hash value was generated using the item name and version for the unique ids of tools/scripts and databases. An autogenerated docker image containing all required R packages and the backend web service of BioInstaller have been deposited at the DockerHub (https://hub.docker.com/r/bioinstaller/bioinstaller).

GitHub API and custom values/functions for querying of version

The querying of versions of bioinformatics tools/scripts and databases of a GitHub or non-GitHub project is the basic function of BioInstaller. For GitHub projects items, the GitHub APIs were used to access the projects version information, such as release, tags, and branches. All released versions will be used as the available versions and returned to BioInstaller (Figure S1a). However, the situation becomes more complicated if the resources have not been published on GitHub. Here, we propose two types of methods of parsing item versions. Method I: If the released versions are fixed, users can write it in the ‘version_available’ field in the configuration file. Method II: Utilizing the R packages rvest (https://CRAN.R-project.org/package=rvest) and RCurl (https://CRAN.R-project.org/package=RCurl) (Wu et al. 2016), we established an R functions pool to dynamically query the version of items from the original release website (Dataset S1). The demo function to query the latest version of GMAP is shown in Figure S1b. This is useful for automating a pipeline to build the precompiled binary version.

Mirror resource for an invalid link

Network transferring is a common problem in bioinformatics data analysis. A mirror resource is one option to partially resolve these problems, including an invalid link and network blocking. BioInstaller allows users to set any numbers of mirror URLs for their tools/scripts and databases.
to avoid the possible problems caused by network transmission. As shown in Figure S1c, the mirror URLs of Miniconda (https://conda.io/miniconda.html) are separately provided by the official and our hosts. Notably, established mirror URLs of bioinformatics resources can be used in the spack (Gamblin et al. 2015) and other similar tools to build the cache files.

**TOML format configuration files**

Massive bioinformatics tools/scripts and databases have been integrated into BioInstaller. Tom's Obvious, Minimal Language (TOML) is a popular and human-readable configuration formats supporting comments. We uses standard TOML format configuration file to store required information of the included bioinformatics tools/scripts and databases. These configuration files can be reused in other bioinformatics software packages or data-analysis pipelines via online accession or as a file copy. We have provided six directories to store different types of TOML files including ‘github’, ‘nongithub’, ‘database’, ‘web’, ‘docker’, and ‘shiny’. Due to the broad compatibility of BioInstaller, any resource published on docker, GitHub, Zenodo (https://zenodo.org/) or other platforms can be supported.

**Implementation of the Shiny application**

To increase the convenience of BioInstaller for nonprogramming users, a user-friendly web application was developed based on Shiny (Chang et al. 2015). The user interface (UI) of BioInstaller was constructed using the R package shinydashboard (https://cran.r-project.org/package=shinydashboard) and Shiny (Chang et al. 2015). Output tables were generated by the R package DT (https://CRAN.R-project.org/package=DT) and wrapped JavaScript library DataTables (https://datatables.net/). Charts were mainly generated by published R packages and in-house scripts or R packages that all support interactive update and export of PDF, SVG, and PNG format plots. The tab items of the BioInstaller Shiny application at the left side of the navigation bar can be used to switch among all available modules, including ‘Introduction’, ‘Dashboard’, ‘Upload’, ‘File Viewer’, ‘Pipeline’, ‘Instant’, ‘Installer’, and ‘Setting’. The detail usage guidelines are provided on our host (http://bioinfo.rjh.com.cn/labs/jhuang/tools/BioInstaller/), and R users can also use the browser vignettes functions in R to access these documents.

**RESULTS**
Overview and practices of BioInstaller’s functionalities

A comprehensive R package was developed that could be used to quickly construct interactive and reproducible biological data analysis applications based on the R platform (Figure 2). The functionalities (Table 1, Dataset S2) of BioInstaller were divided into six parts based on whether users use BioInstaller or not: 1) deployment of resources, 2) collection of resources, 3) sharing of resources, 4) construction of pipelines, 5) construction of Shiny applications, and 6) reproducible data analysis. An example of a real project (annovarR, https://github.com/JhuangLab/annovarR, under development) is shown in Figure 2 to illustrate the full workflow for BioInstaller utilization, which was designed to integrate various genetic variant annotation and visualization tools, including public command line tools, R packages and custom annotation and visualization functions. Using the code library, predefined TOML files (database resources and plugins), and the docker file of BioInstaller, we could easily customize the BioInstaller-established Shiny application to work on the genetic variants annotation tasks. If BioInstaller is not used, we need to develop the UI and server code of the Shiny application for a large number of universal functions, such as the file management system, background task submission and queue management, and tracking of the output log and files. The docker image of BioInstaller is also out-of-the-box and could be modified and applied to our own works. Based on the integrated installer (e.g., conda, spack, and BioInstaller) and simplified TOML files of BioInstaller, users can collect, share, and deploy genetic variant annotation databases and tools with one-stop service. As a real practice of BioInstaller, we collected and shared more than 157 tools/scripts and 110 databases (Table 2, S1, S2) in the configuration pool of BioInstaller, including genetic variant annotation databases and tools; the meta information is freely available and hosted on the public GitHub website (https://github.com/JhuangLab/BioInstaller/tree/master/inst/extdata/config). The raw files are stored on the original websites (e.g., https://github.com, https://sourceforge.net/, http://annovar.openbioinformatics.org/, etc.) and our host.

Comparison of BioInstaller with existing tools for the collection and sharing of bioinformatics resources

To better understand the advance provided by BioInstaller in terms of the collection and sharing of bioinformatics resources, we further compared BioInstaller with several existing tools, including Omictools (Henry et al. 2014) and Datasets2Tools (Torre et al. 2018) (Table 1, 2), the
two most comprehensive meta databases focused on bioinformatics tools. All provide a web forum
to update the meta database of bioinformatics resources. However, BioInstaller offers an off-line
way to develop the users’ own meta databases via an unlimited configuration file pool ( TOML and
SQLite format) that is easy to carry and share and is independent of programming knowledge. In
addition, the developed R functions and Shiny application can be used to query and download the
linked or isolated file databases, such as appendix data from papers, annotation databases for
genetic variants, genome sequences, etc. In most cases, it is suitable to tightly combine the meta
database with the file database. Therefore, we designed and shared an upload module in the Shiny
application to set the meta information for all files, and users can add the description, genome
version, custom file types, and other customizable fields. Both Omictools and Dataset2Tools only
include the items in their databases and do not integrate external resources. BioInstaller not only
can be used to collect users own resources, but also can be used to integrate external resources.

Summary of supported bioinformatics tools/scripts and databases

For now, 157 tools/scripts and 110 databases are natively supported in BioInstaller (Figure 1,
Table 2, S1, S2). First, we covered the most commonly used tools in each bioinformatics analysis
process, including data quality control (n=17), alignment and assembly (n=27), variant detection
(n=32) or annotation (n=12), high-throughput sequence (HTS) manipulation (n=17) and
visualization libraries (n=11) (Table 1, S1), etc. Second, BioInstaller also provides abundant
databases for annotating data or satisfying software dependencies. With BioInstaller, users can
easily download UCSC sequence and annotation data (n=4995) (Dataset S3), blast databases
(n=29) (Table S2), allele frequency databases (n=17), variant effect prediction databases (n=29),
and disease-related (n=13), drug-related (n=4), noncoding region-related databases (n=15) (Table
2, S2), among others. Notably, we collected and constructed 20 genetic variant annotation
databases, which can be directly used in other variants annotation tools, including ANNOVAR
(Wang et al. 2010), vcfanno (Pedersen et al. 2016), and annovarR
(https://github.com/JhuangLab/annovarR).

BioInstaller has been released on CRAN for one and a half years and has accumulated a certain
number of users, with a total of 19,912 downloads from CRAN (2018.8.3). In the recent release
(v0.3.5), we provided the Shiny application and significantly expanded the supported tools/scripts
and databases. The number of supported tools/scripts and databases is still increasing and is being
applied to other related projects, such as the integrated genetic variants annotation tool annovarR (https://github.com/JhuangLab/annovarR).

Examples of BioInstaller R functions

We have demonstrated the basic structure, functions, and web service of BioInstaller. The full help document is available at http://bioinfo.rjh.com.cn/labs/jhuang/tools/BioInstaller/articles/. Because most of the Shiny application UIs are wrapped with R functions, we use several use examples to illustrate the R functions of BioInstaller.

Example #1: Install packed or unpacked bioinformatics tools. We use the Ion Torrent Variant Caller (TVC) (Zook et al. 2014) and svaba (Wala et al. 2018) to show how to install or download the bioinformatics tools or scripts that are not supported by other package management tools.

```r
> library(BioInstaller)  # Library the R package
> set.biosoftwares.db(“~/.BioInstaller/info.yaml”)  # Store the installation information
> install.bioinfo(show.all.names = TRUE)  # Get all items name supported by BioInstaller
> install.bioinfo(name = “tvc”, show.all.versions = TRUE)  # Get all available versions of tvc
> install.bioinfo(name = “svaba”, show.all.versions = TRUE)  # Get all available versions of svaba
> install.bioinfo(name = “tvc”, download.dir = “/path/tool/tvc”)  # One-click install the tvc
> install.bioinfo(name = “svaba”, download.dir = “/path/tool/svaba”)  # One-click install the svaba
> show.installed()  # Get all installed tools
> get.info(“svaba”)  # Get the svaba installation information, such as update time and version
```

Example #2: Download genetic variants annotation databases. Genetic variants annotation is a common and high-demand task for most biomedicine research, especially for examining the correlations between phenotype and molecular features, such as germline and somatic mutations. The followed example describes how to download the genetic variants annotation databases dbSNP, CIViC, DisGeNET, and CancerHotspot (Chang et al. 2016; Griffith et al. 2017; Piñero et al. 2017).

```r
> install.bioinfo(“db_annovar_avsnp”, extra.list = list(buildver = “hg19”), download.dir = “/path/db/”)  # install the latest dbSNP from ANNOVAR website
```
> crawl.all.versions("db_annovar_avsnp") # Download all dbSNP to current directory
> install.bioinfo("db_civic", download.dir = "/tmp/db") # Download the nightly version of CIViC
database
> install.bioinfo("db_disgenet", download.dir = "/tmp/db") # Download the DisGeNET database
> install.bioinfo("db_cancer_hotspots", download.dir = "/tmp/db") # Download the DisGeNET
database β

**Example #3:** Download an annotation database based on the supplementary files of published
papers. The followed example is an epigenetic genes classification (e.g., reader, writer, eraser)
database only available in the papers supplementary file (Huether et al. 2014).

> install.bioinfo("db_annovar_epi_genes", extra.list = list(buildver = "hg19"), download.dir =
"/path/db") # install the epigenetic genes database from our website

**User-interfaces and functions of the Shiny application**

**Introduction module.** Utilizing the Shiny function ‘includeMarkdown’, we generated the
‘Introduction’ module page from Markdown, a lightweight markup language, format document
(Figure S2a).

**Dashboard module.** The ‘Dashboard’ module includes the system monitors, such as hardware
(Disk and memory), queue tasks, task log, installed R packages, Python packages, conda
environments, and the other information of the operating environments (Figure S2b, c, S3). The
monitored data stream is automatically updated once every 10 seconds (Figure S2b). A demo table
output in the dashboard lists all files in the environment variable ‘PATH’, where users can use the
selector at the lower left quarter to customize the row numbers (5, 10, 25, 50, and all) (Figure S2c). All output tables in BioInstaller can be easily exported to CSV, XLS, PDF files or directly
copied to the clipboard. Monitor plugins related to the information of the R system (Figure S3a,
b, c), BioInstaller (Figure S3d), conda (Figure S3e) and spack (Figure S3f) are integrated in this
work, which can reduce user input of extra command line commands and facilitate sharing with
others.

**Upload module.** The ‘Upload’ module is used to upload files to the BioInstaller Shiny web
platform. Optional fields, such as file type, genome version, and description, can be stored in the
SQLite format database with the uploaded files path and the files md5 value (Figure S2d). When
uploading a file, users need to click the ‘Save’ button to confirm the upload behavior and update
the database (Figure S2e). Before the confirmation click, users can preview the file and make a
final decision (Figure S2f). Files with sizes ranging from 0.25 GB to 8 GB were tested on the
Shiny application (Table S3). For files larger than 10 GB, we recommend using the rsync or FTP
service to transfer files and then adding the corresponding description and records in TOML or
SQLite databases.

**File viewer module.** The ‘File viewer’ module is used to manage all uploaded files in the
BioInstaller Shiny application that supports view, delete and download, and all files can be used
in the other plugins of the BioInstaller Shiny application, mainly in the ‘Pipeline’ and ‘Instant’
modules (Figure 3a, b).

**Pipeline module.** The ‘Pipeline’ module is used to integrate complicated bioinformatics analysis
workflows or other small scripts. An in-house interpreter R function was used to parse the plugin
configuration files to generate the Shiny UI and server functions. A small script creating a data
analysis directory structure was used as the demo for ‘Pipeline’ (Dataset S4). Users can input the
project name and the parent directory to create a series of predefined directories. The R commands
used in this task are editable at the bottom of the box (Figure 3c). After users click the ‘Submit’
button, BioInstaller will generate a random character as the submitted task key. Users can use this
key to retrieve the output information, such as files and logs, in the ‘Dashboard’ module (Figure
3c, d, e, f). All submitted tasks enter the task queue supported by the SQLite database using the R
package litseq (https://CRAN.R-project.org/package=liteq). Tasks in the queue are automatically
checked by the activated workers (Figure 3e, f).

**Instant module.** The ‘Instant’ module is used to run the real-time plots and data analysis, and
similar to the ‘Pipeline’ module, the UI and server were automatically generated via plugin
configuration files (Dataset S5). We used the meta database query of BioInstaller, Datasets2tools
(Torre et al. 2018), PubMed, and plots of Maftools (Mayakonda & Koeffler 2016), a cancer
somatic mutations visualization tool, as the demo to demonstrate the function. Users can select the
input files defined in the plugins configuration file (TOML) or user-uploaded files. The commands
are stored in the bottom of the boxes and can be modified by the user. After clicking the ‘Run’
button, all output box codes, such as output plots and tables, run on the server side and are returned
in real time to the Shiny UI (Figure 4). We developed several plugins to query and access several
meta databases related to bioinformatics, such as the BioInstaller meta databases (Figure 4a, b),
Datasets2tools (Torre et al. 2018) (Figure 4c, d), and PubMed (Figure 4e, f). The powerful visualization functions of R packages are also supported in the ‘Instant’ module. As shown in Figure 4g, h, users can obtain the demo output (PDF and PNG format) of Maftools. After running all box codes, a single box can be separately updated and exported by users.

**Installer module.** The ‘Installer’ module is the main Shiny interface of BioInstaller for downloading and installing bioinformatics tools/scripts and databases. We provide the Shiny interfaces of BioInstaller, conda and spack (Figure S4). The ‘Installer’ module is similar to the ‘Pipeline’ module, which is also needed to submit a task to the queue. The status and log information can be retrieved in the ‘Dashboard’ module. Three basic use cases of the BioInstaller Shiny application are available: 1) download db_annovar_refgene database (Figure S4a); 2) create conda environment (Figure S4b); 3) install ‘zlib’ using spack (Figure S4c).

**Setting module.** The setting module is the interface for setting the value of the variable used in the BioInstaller plugins or R files. Both a Shiny UI and a YAML editor are offered for users (Figure S5a, b). Any updates of the YAML editor (Figure S5b) can change and refresh the Shiny UI options (Figure S5a). It is helpful for users to manage various material related to BioInstaller and its plugins.

In most cases, through the one-click interface of BioInstaller, users can easily download and install the desired bioinformatics resources without any command line skills. Functions for automatic compiling from the source file with the dependent software or database are also supported in BioInstaller. However, for complicated software with high system dependence, we recommend using the interfaces of conda (https://conda.io/docs/) and spack (Gamblin et al. 2015).

**Portable message queue for background tasks based on SQLite**

Tasks with long-time costs are challenging in Shiny, which always blocks the other interactive operations simultaneously when the previous task has not been finished. Here, we utilized the R package litseq (https://CRAN.R-project.org/package=liteq) to submit and manage the background queue tasks. litseq is portable and lightweight. litseq does not require extra software or service from other programming platforms and can work on any clusters server running computing-intensive tasks. The developed queue worker in BioInstaller can be used for all other background tasks submitted by litseq. All litseq-submitted tasks of BioInstaller are assigned a unique identification id. All executed commands, output logs, and others are saved in the permanent files.
Opencpu backend service improves reproducibility

Opencpu (Ooms 2017) is an R package for reproducible research that can expose a web REST API interface with R, Latex and, Pandoc. The R functions of BioInstaller are invoked by the activated Opencpu R process or daemon service. For other programming platform users, this is one possible method for utilizing the R functions of BioInstaller (Figure 1). The output of JSON and text formats are returned when using the browser access (Figure 5a) or simulated requests. Three of the most basic APIs usages of BioInstaller were used to demonstrate how it works: 1) obtaining all supported tools/scripts and databases; 2) acquiring available versions of the appointed item; 3) installing a tool in a directory (Figure 5b). Notably, a random string, such as “x0a469794fa”, will be generated as the key of Opencpu to obtain the output of one R session. Both JSON and text format output can be returned by Opencpu backend APIs (Figure 5c).

Docker container of BioInstaller

A prebuilt docker image is available on the docker hub (https://hub.docker.com/r/bioinstaller/bioinstaller), and the latest code change of the BioInstaller repository can automatically trigger an update of the docker image. In the docker image, we integrated and configured three types of web services, including Opencpu, Shiny (Chang et al. 2015; Ooms 2017), and the RStudio server (https://www.rstudio.com/products/rstudio-server/). The followed commands can be used to deploy and start the service of BioInstaller service.

$ docker pull bioinstaller/bioinstaller
$ docker run -it -p 80:80 -p 8004:8004 bioinstaller/bioinstaller

Users can deploy a new instance host of BioInstaller and all other web services in a few minutes, and other tools/scripts and databases are also allowed to be embedded in this docker image using the Dockerfile (https://github.com/JhuangLab/BioInstaller/blob/master/Dockerfile).

Use the GitHub forum to share, rate, and discuss the bioinformatics resources

The full-text search is natively supported by the GitHub website with highlight and age forwarding functions (Figure S6a, b). To simplify the submitting of new items to BioInstaller, the GitHub repository issues page (https://github.com/JhuangLab/BioInstaller/issues) is recommended for other users to share, rate, and discuss bioinformatics tools/scripts and databases with a designated
358 label (Figure S6c). The ‘watching’ function of GitHub can allow users to receive notifications of
359 all conversations on the BioInstaller. Another advantage of establishing a free sharing community
360 based on the GitHub is that all history changes on the code and forum posts can be recorded and
361 retrieved. A rating function for bioinformatics tools/scripts or databases is also feasible by
362 calculating the points corresponding to thumbs up or down.

363 DISCUSSION

364 Bioinformatics tools/scripts and databases are widely used in various data analysis projects. The
365 construction of interactive and reproducible biological data analysis applications is critical for
366 most bioinformatics data analyses (Henry et al. 2014; McQuilton et al. 2016; Ohno-Machado et
367 al. 2017). The integrative utilization of these resources is becoming increasingly important for
368 improving integrated biosciences data analysis. R language, as the most popular programming
369 language for statistics, biological data analysis, and big data, has provided massive useful R
370 packages for various data analysis efforts, especially the NGS field. However, there has been no
371 comprehensive and free R application that can support file upload and management, perform long-
372 time computation with a tasks submission system, track and record the output of files and log,
373 develop extendible plugins, add or remove functions of the application in real time, and respond
374 for REST APIs. Another common problem for users of R and other programming platforms for
375 biological data analysis is that massive bioinformatics resources are isolated and scattered, which
376 significantly increases the difficulty of deploying, collecting and sharing these resources. Well-
377 known software distribution tools that do not need root privileges, such as conda
378 (https://conda.io/docs) and spack (Gamblin et al. 2015), were designed for comprehensive fields
379 and usually lack support for life science resources. Bioconda is a fine example of the centralized
380 installation of bioinformatics software (approximately 1900 items) that has significantly improved
381 the reproducibility of bioinformatics data analysis (Gruning et al. 2018). However, this is not
382 sufficient compared with the rapid increase in software and databases in the life sciences field.
383 As described in this study, we present a comprehensive, free and open-source platform,
384 BioInstaller, to construct the interactive and reproducible biological data analysis applications.
385 BioInstaller contains the R functions, the Shiny application, REST APIs and the docker image.
386 This platform and the practices described in this work are sufficient for most R users to
387 conveniently and quickly develop an interactive and reproducible biological data analysis
application with diverse predefined functions (e.g., file management, task submission, plugin management system, logging, etc.), plugins, and files offered by BioInstaller. Moreover, based on the TOML format files, we have also integrated hundreds of bioinformatics resources required for the wide field of bioinformatics, such as sequence alignment, variant calling and annotation, and so on. We hope this newly presented open source platform for R users can reduce the difficulty of constructing the interactive and reproducible biological data analysis applications and further improve the interactivity and reproducibility of bioinformatics data analysis.

CONCLUSION

As described in this work, we established a new platform to construct interactive and reproducible biological data analysis applications based on R language. This platform contains diverse user interfaces, including the R functions and R Shiny application, REST APIs, and support for collecting, managing, sharing, and utilizing massive bioinformatics tools/scripts and databases.

ACKNOWLEDGEMENT

We would like to thank the many BioInstaller users who provided feedback and suggestions. We thank the Centre for HPC at Shanghai Jiao Tong University for supporting computing platforms.

REFERENCES


Consortium GT. 2013. The Genotype-Tissue Expression (GTEx) project. *Nat Genet* 45:580-585. 10.1038/ng.2653


10.1093/nar/gkv1189


10.1093/nar/gkw943


10.1093/nar/gkw1134


Figure 1 (on next page)

Overview of structure and functions of BioInstaller.

Bioinformatics tools, scripts and databases are supported by BioInstaller. Bootstrap and Shinydashboard are used to construct the front-end interface. The R functions, Shiny and OpenCPU services and the SQLite and TOML databases were applied in the back-end.
Figure 2 (on next page)

The relevance, applicability and a real example of BioInstaller
Non-programmer

Meta database

File database

GitHub
Zenodo
CRAN
Bioconductor
Conda
Spack
BioInstaller

Pre-defined functions and files (BioInstaller)

GitHub
Zenodo
Pubmed
Datasets2Tools
BioInstaller

Programmer

R function, Shiny application, REST APIs

Construction of pipelines

Deployment of resources

Collection of resources

Sharing of resources

Construction of Shiny application

Reproducible data analysis

Users

Resources types

Service provider

Applications

Related to genetic variants annotation

License required
Local network required

Meta database
(Extended BioInstaller style TOML files)

Annotation databases
Annotation tools

File database
(Deposit on multiple web servers)

Get formatted meta information
(BioInstaller style TOML files)

Construction of integrative R package to annotate the genetic variants

Shiny application
(BioInstaller style plugins)

Related to genetic variants annotation

Public + In-house

ANNOVAR, VEP, AnnotationDbi, vcfanno, UCSC, Papers supplementary

URLs
Version
Genome versions

Construction of Shiny application

R function, Shiny application, REST APIs

License required
Local network required

ANNOVAR, VEP, AnnotationDbi, vcfanno, UCSC, Papers supplementary

Construction of pipelines

Deployment of resources

Collection of resources

Sharing of resources

Construction of Shiny application

Reproducible data analysis

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.27221v1 | CC BY 4.0 Open Access | rec: 17 Sep 2018, publ: 17 Sep 2018

custom annotation functions (R)
Figure 3 (on next page)

Shiny application modules of file viewer and pipeline.

(A) Uploaded files are showed on the table where view, deletion, and download function are provided. (B) The interface of the preview result. (C) Easy project was used as the demo in pipeline module, which could be used to create a series of directories via submitting a queue task with two parameters: project name and parent directory. (D) The dialog box displays a prompt message with a queue character key. (E) Task queue and queue information can be requested by the character key in the dashboard module. (F) Function to get the output log of the submitted task.
**Figure 4** (on next page)

Shiny application 'Instant' module

(A)(B) A demonstration show how Shiny 'Installer' module works by downloading 'db_ucsc_refgene'. Dynamic and interactive manipulations are supported. Log information of submitted download/install job can be recalled using given random characters. (C)(D) The input box and output log when a new softwares environment is created by the conda plugin of 'Installer' module. (E)(F) The input box and output log when 'zlib' is installed by the spack plugin of 'Installer' module.
BioInstaller meta database query

Query type

- github

Run

Dataset2tools meta database query

Query type

- tool

Key Words

Disease name

Canned analysis Cession

Genset

Counts

100

Run

Pubmed query

Term (e.g. 30105797,30105798,B-ALL)

vcfanno

Counts

100

Pubmed Account Key

1931124975d1e7f36d15036d31e3bfe09

Run

Output of bioinstaller (Table)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Publication</th>
<th>resource.db</th>
</tr>
</thead>
<tbody>
<tr>
<td>vcfanno</td>
<td>allows you to quickly annotate your VCF with any number of INFO fields from any number of VCFs or BED files. It uses a simple config file to allow the user to specify the source annotation fields and fields and how they will be added to the info of the query VCF.</td>
<td>PeerJ Preprints</td>
<td><a href="https://doi.org/10.7287/peerj.preprints.27221v1">https://doi.org/10.7287/peerj.preprints.27221v1</a></td>
</tr>
</tbody>
</table>

Output of Dataset2tools (Table)

<table>
<thead>
<tr>
<th>analysis</th>
<th>tool_description</th>
<th>articles</th>
<th>tool_name</th>
<th>tool_accession</th>
</tr>
</thead>
<tbody>
<tr>
<td>vcfanno</td>
<td>Functional annotation of genomic variants from high-throughput sequencing data</td>
<td><a href="https://doi.org/10.1093/nar/gkp063">https://doi.org/10.1093/nar/gkp063</a></td>
<td>ANNOVAR</td>
<td>DCT0003217</td>
</tr>
</tbody>
</table>

Output of pubmed (Table)

<table>
<thead>
<tr>
<th>PMID</th>
<th>DOI</th>
<th>Title</th>
<th>Abstract</th>
<th>PublishDate</th>
<th>Journal</th>
</tr>
</thead>
<tbody>
<tr>
<td>77210555</td>
<td>10.1386/1372055</td>
<td>Vcfanno: fast, flexible annotation of genomic variants.</td>
<td>The integration of genome annotations is critical to the identification of genomic variants that are relevant to studies of disease or other traits. However, comprehensive variant annotation with diverse file formats is difficult with existing methods. Here we describe vcfanno, which helps identify and summarises attributes from multiple annotation files and integrates the annotations within the INFO column of the original VCF file. By leveraging a parallel &quot;first-matching-keeping&quot; algorithm, we demonstrate substantial performance gains by annotating ~10,000 variants per second with 10 attributes from 17 commonly used genome annotation resources. vcfanno is available at <a href="https://github.com/tanglab/vcfanno">https://github.com/tanglab/vcfanno</a> under the MIT licence.</td>
<td>2016-09-01</td>
<td>Genome Biology</td>
</tr>
</tbody>
</table>

Output of maftools MAF summary

R commands for task

```
# R code for task
# input files: sample.maf, sample.txt, sample_summary.txt
# analysis: addSample = TRUE
# output files: sample.maf, sample_summary.txt
```

Export | Update plot

<table>
<thead>
<tr>
<th>height (cm)</th>
<th>width (cm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>500</td>
</tr>
</tbody>
</table>
Figure 5 (on next page)

REST APIs of BioInstaller.

(A) Workflow of REST APIs of BioInstaller that JSON and TEXT returns through the GET/POST query. (B) Using curl to invoke background R functions of BioInstaller. (C) The key character with GET method is provided to get the background R session output.
**A**

- **GET/POST**
- **Opencpu**
  - JavaScript
  - R
- **BioInstaller**
  - Search
  - Download
- **Backend**
  - SQLite
  - TOML files

**B**

```bash
curl http://localhost:5656/ocpu/library/BioInstaller/R/install.bioinfo -d "show.all.names=TRUE" -X POST

Show all items supported by BioInstaller

curl http://localhost:5656/ocpu/library/BioInstaller/R/install.bioinfo -d "name='bwa', show.all.versions=TRUE" -X POST

Show all versions of Bwa


Download and install latest Bwa

curl http://localhost:5656/ocpu/library/BioInstaller/R/install.bioinfo -d "name='db_annovar_1000g', destdir='/opt/annovardb', extra.list=list(buildver='hg19')" -X POST

Download 1000 Genome Project annotation database
```

**C**

```bash
curl http://localhost:5656/ocpu/tmp/{key}/R/.val/json

Get the JSON format value of returned output

curl http://localhost:5656/ocpu/tmp/{key}/R/.val/text

Get the text format value of returned output

curl http://localhost:5656/ocpu/tmp/{key}/R/.val/print

Get the function ‘print’ output of returned value
```
**Table 1** (on next page)

List of the relevance and applicability of BioInstaller
<table>
<thead>
<tr>
<th>Deployment of resources</th>
<th>With BioInstaller</th>
<th>Without BioInstaller</th>
</tr>
</thead>
<tbody>
<tr>
<td>User-interfaces</td>
<td>R functions, Shiny UI, REST APIs (Conda, Spack, and other tools/scripts)</td>
<td>Command-line tools (Conda, Spack, and custom tools)</td>
</tr>
<tr>
<td>Retrieve installed packages</td>
<td>Integrated Shiny dashboard page including R packages, conda and Python packages, Spack packages, and BioInstaller resource</td>
<td>Multiple command line operations</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Collection of resources</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Local development</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Need to register an account</td>
<td>Not need</td>
<td>Need</td>
</tr>
<tr>
<td>Type of backend databases</td>
<td>Default use TOML and SQLite (potable purpose) Plugins for other types</td>
<td>MySQL</td>
</tr>
<tr>
<td>Resources hosts</td>
<td>No limitation</td>
<td>Centralized</td>
</tr>
<tr>
<td>File sizes</td>
<td>No limitation</td>
<td>Limited</td>
</tr>
<tr>
<td>PubMed query</td>
<td>Integrated R codes with secret key (no limited access) Shiny UI with formatted table</td>
<td>Isolated R codes without secret key (limited access, n&lt;=20) Online version without formatted table</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Sharing of resources</th>
<th></th>
<th>Form or configuration file required more skills</th>
</tr>
</thead>
<tbody>
<tr>
<td>Medium</td>
<td>Simplified TOML format files</td>
<td></td>
</tr>
<tr>
<td>Download service</td>
<td>Local Shiny application</td>
<td>Centralized web service or command line tools</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Construction of pipelines</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Store of meta information (e.g. URL and version)</td>
<td>Pre-defined TOML file</td>
<td>De novo source code (e.g. ANNOVAR and fusioncatcher)</td>
</tr>
</tbody>
</table>

<p>| Construction of Shiny application | |
|-----------------------------------| |</p>
<table>
<thead>
<tr>
<th>Pre-defined pages</th>
<th>Pre-defined Shiny UI and server (Dashboard, file management, task submission, logging, export and update of plots, exception handling, setting)</th>
<th>Isolated examples UI and server codes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Difficulty</td>
<td>Easy to construct the Shiny application (Plugins + optional R codes)</td>
<td>Relatively complicated (Require R codes for UI and server)</td>
</tr>
<tr>
<td>Reproducible data analysis</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Logging</td>
<td>Support</td>
<td>Manual</td>
</tr>
<tr>
<td>Docker image</td>
<td>Pre-defined docker image with Shiny, Rstudio, and Opencpu services</td>
<td>Most not</td>
</tr>
</tbody>
</table>
Overview comparison of BioInstaller and existing tools on the collection and share of bioinformatics resources
<table>
<thead>
<tr>
<th>Infrastructure and Utilities</th>
<th>BioInstaller</th>
<th>Omictools</th>
<th>Datasets2Tools</th>
</tr>
</thead>
<tbody>
<tr>
<td>Programing language</td>
<td>R, JavaScript</td>
<td>HTML/CSS/JavaScript</td>
<td>HTML/CSS/JavaScript</td>
</tr>
<tr>
<td>Chrome extension</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Web service</td>
<td>R Shiny</td>
<td>Web</td>
<td>Web</td>
</tr>
<tr>
<td>R functions</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>REST APIs</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Backend database</td>
<td>TOML and SQLite</td>
<td>Not available</td>
<td>MySQL</td>
</tr>
<tr>
<td>Docker image</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td><strong>Functionality</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Access and collect meta database</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Access and collect file database</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Integration of external resources</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>PubMed query</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Dataset query</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Number of supported resources</td>
<td>Integrated</td>
<td>High</td>
<td>Medium</td>
</tr>
<tr>
<td>Version query</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Download service</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Local branch and development</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td><strong>Input and output</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>R functions, Web text, APIs</td>
<td>Web text only</td>
<td>Web text + APIs</td>
</tr>
<tr>
<td>Output</td>
<td>Text, table, plots, and Web page (PNG, SVG and PDF)</td>
<td>Web page</td>
<td>Text and Web page</td>
</tr>
</tbody>
</table>
Table 3 (on next page)

Summary of BioInstaller included tools/scripts and databases
<table>
<thead>
<tr>
<th>Category</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1, Tools/scripts</strong></td>
<td></td>
</tr>
<tr>
<td>Alignment and assembly</td>
<td>27</td>
</tr>
<tr>
<td>Quality control</td>
<td>17</td>
</tr>
<tr>
<td>HTS manipulation</td>
<td>17</td>
</tr>
<tr>
<td>Association analysis</td>
<td>6</td>
</tr>
<tr>
<td>Genetic variants annotation</td>
<td>12</td>
</tr>
<tr>
<td>Detection of SNVs, INDELs and SVs</td>
<td>32</td>
</tr>
<tr>
<td>Immunity-associated</td>
<td>2</td>
</tr>
<tr>
<td>Isoform analysis</td>
<td>3</td>
</tr>
<tr>
<td>Gene expression analysis</td>
<td>9</td>
</tr>
<tr>
<td>Network analysis</td>
<td>3</td>
</tr>
<tr>
<td>Visualization libraries</td>
<td>11</td>
</tr>
<tr>
<td>System dependence</td>
<td>18</td>
</tr>
<tr>
<td><strong>2, Databases</strong></td>
<td></td>
</tr>
<tr>
<td>Variant-level</td>
<td></td>
</tr>
<tr>
<td>Allele frequency</td>
<td>17</td>
</tr>
<tr>
<td>Variants Effect prediction</td>
<td>29</td>
</tr>
<tr>
<td>Disease-related</td>
<td>6</td>
</tr>
<tr>
<td>Gene-level</td>
<td></td>
</tr>
<tr>
<td>Basic information</td>
<td>8</td>
</tr>
<tr>
<td>Gene function</td>
<td>3</td>
</tr>
<tr>
<td>Disease-related</td>
<td>7</td>
</tr>
<tr>
<td>Drug related</td>
<td>4</td>
</tr>
<tr>
<td>Noncoding RNA related</td>
<td>15</td>
</tr>
<tr>
<td>Reference genome</td>
<td>9</td>
</tr>
<tr>
<td>Protein related</td>
<td>4</td>
</tr>
<tr>
<td>Others</td>
<td>8</td>
</tr>
</tbody>
</table>