Toward the commoditization of translational genomic research: Design and implementation features of the Galaxy genomic workbench.

Ross Lazarus, MBBS, GDipCSci\(^1\), James Taylor PhD\(^2\), Weiliang Qiu PhD\(^1\), Anton Nekrutenko PhD\(^3\)
\(^1\)Harvard University, Boston, MA; \(^2\)Columbia University, NY; \(^3\)Penn State University, PA.

Abstract
Although there is now plenty of genomic data and no shortage of analysis methods for translational genomic research, many biologists do not have efficient and transparent access to the computational resources they need. No single data resource or analysis application is ever likely to efficiently address all aspects of any individual researcher’s needs, so most researchers are forced to manually integrate data and outputs from multiple resources. The inevitable heterogeneity of data formats and of command syntax between data resources and software applications presents a major obstacle, particularly to those biologists lacking practical informatics skills. We describe some design and implementation features of an open-source application that supports the integration of the best available third-party genomics software applications, data and annotation resources into a coherent framework, substantially overcoming many practical challenges associated with actually doing translational genomic research.

Introduction
A torrent of new genomic sequences\(^1\), nearly a billion validated genotypes from the International HapMap project\(^2\), billions of genotypes from large publicly funded studies available through dbGap\(^3\), together with recent advances in technologies for sequencing, genotyping and gene expression measurement, have created enormous opportunity, as well as something of a crisis, for translational research involving genomic aspects of human disease. Although there are many useful and powerful resources for annotation, data management and analyses capable of operating with specific groups of available data, they tend to exist as stand-alone “silos”, mostly with mutually incoherent data formats, not always scaling well to very large data collections, and not available in any single integrated framework. Research that integrates across multiple tools usually requires specialized informatics skills and effort. Researchers without access to informatics support may be effectively starved of these resources despite the current glut.

Speed bumps for the roadmap
Substantial resources are being invested in the accumulation of vast repositories of expression, genetic and genomic data, and in the creation of powerful, community supported software capable of analyzing it. Despite these exciting and valuable accomplishments, the fraction of biologists who can efficiently incorporate all these resources into their daily research practice today, remains relatively small, partly because data repositories and software packages have heterogeneous interfaces and formats, requiring substantial effort on the part of each user to transform data from any one data repository or application output into the specific formats needed for further processing in a translational research workflow.

There is no shortage of data or application software for analyzing it. The Internet exposes vast data repositories and software developer web sites distributing free and community supported analysis software. Rather, the central practical challenge is inconsistency in software interfaces and data formats. Those with access to appropriate informatics skills can create new software infrastructure to construct reproducible pipelines to transform data, and integrate all these rich resources into their work. However, the majority of biologists are effectively limited in their access to many new and important methods and data for want of access to the necessary enabling skills.

Heterogeneity and transparency
Wrestling data from one resource format to the formats required by various applications, and combining the heterogenous outputs from multiple analysis packages is nearly always required during genomic research synthesis. Unfortunately, this is often done with interactive spreadsheet software – the bioinformatic equivalent of string and duct tape, leading to a second major problem – lack of reproducibility for some important research steps.

Translational research demands transparency. Reproducibility is necessary for transparency. Other
scientists will be hard to convince if an analysis cannot be readily reproduced. Data and applications must be coherently integrated, in a way that supports reproducible research, requiring little or no wasted investigator effort in painstaking manual reformatting of data or analytic results. Each step of analysis must be recorded in a way that permits reproducing the process precisely.

Mega-application homogeneity

One approach to solving the heterogeneity challenge is to create a single, huge, all-encompassing translational genomic research application, with data format homogeneity built-in by design. Could any single researcher or even any well-resourced, competent large group of researchers understand or describe, let alone efficiently and correctly implement and integrate, all potentially useful data resources and analysis applications? Modern genomic biology is already far too complex for that, and constantly growing, so the moment the design and implementation of the behemoth application was complete, it would likely already be outdated by the development of some new analysis algorithm or new data repository.

The Galaxy genomics workbench

Is there a sustainable approach to the translational challenges of heterogeneity and transparency? Are we clever enough to design and implement an efficient and robust “one size fits all” application that could integrate every kind of genomic data imaginable and perform every conceivable analysis, while supporting reproducible research and requiring little or no training? Using an appropriate design, and with limited, specific choices of domain and expectation, we present evidence that this is at least possible. We describe Galaxy[4], a functional and useful genomic research workbench, hosted at Penn State University, and remotely accessible to any researcher, at no cost, through an ordinary web browser[5]. Galaxy was designed to provide a uniform web interface, integrating multiple, independent external applications, into persistent user workspace.

The meta-server framework

Accepting that no single group can do it all, it seems sensible to design an open, inclusive, agile, generic framework that can be easily connected to heterogeneous data sources, and heterogeneous analysis applications, so that existing and new resources can quickly be accommodated. The term “application meta-server” seems apt to describe this approach – an application server specifically designed to transparently support the integration (from the user’s perspective) of multiple external analysis applications and data sources. As shown below, this approach imposes no restrictions on the implementation of the actual application software, making it highly adaptable. If it takes command line parameters, writes output files, and requires no interaction, it can be wrapped. Below we describe the simple interface that allows data repositories to appear as inbuilt Galaxy resources. In place of the behemoth mega-application, the imaginary Borg[6] of the Star Trek universe serve as an apt model for Galaxy, rapidly and efficiently assimilating new resources and software applications, into a coherent, persistent, research work space.

Design issues for users and uses

Potential users are themselves a challenge in heterogeneity, with varying backgrounds, skills and interests. Galaxy provides an intuitive and consistent tool selection, tool configuration and job control interface that many users find easy to understand. If the tool wraps an application that produces graphics, these appear in the workspace history, and are available for viewing through the web browser interface. In support of research transparency, Galaxy also maintains persistent user-managed workspaces, recording all tool parameter settings, outputs, primary and intermediate data sets.

Many users will be content with the ready-made public Galaxy service to support their research, and lacking the interest and perhaps the skills, will have no desire or need to install, support or extend the framework for themselves. On the other hand, many larger groups, particularly those with data protected by human subjects and privacy concerns, will want to run a private, secured version. Appropriately skilled individuals can easily adapt the tool menu with local analysis tools and data resources of their own. The current Galaxy implementation supports a pure web browser interface, since Internet connected web browsers are usually available to translational researchers. Any substantial computational workload can be offloaded to a high performance computational cluster backend in order to ensure that the Galaxy service is not bogged down by long running computationally intensive tasks. Galaxy is relatively light-weight in terms of technical and hardware requirements, so that installation and configuration of a private instance can be completed in minutes rather than hours or days[7].
Application and resource interface design issues

A core requirement for a meta-server is facilitating functional integration for independent external applications, external services, and external sources of data. An “external application” can be any local non-interactive command line executable object including R, Python, perl, shell or SAS scripts, or any compiled executable. A new Galaxy tool is constructed by creating a document specifying a Galaxy application interface in an eXtended Markup Language (XML) format. A simple syntax specifies the parameters to be provided by the user and optionally, validation rules for their contents, together with the specific application and command line to be executed, optional on-line user documentation and help text, and a functional test for the automated test framework to exercise and report. Multi-page interfaces can easily be described, with user choices from one page being propagated through to subsequent pages, making complex, multi-branching application interfaces relatively easy to construct. Galaxy automatically provides a consistent user interface based on these individual tool definition files, for each application, so they appear natively integrated into the Galaxy workspace, from the user’s perspective.

In terms of integrating external data resources such as the UCSC genome browser[8] table view, Galaxy exposes a very simple interface, easily implemented based on minor changes to the resource’s existing web based interactive interface. The user sees the usual, familiar, web based resource interface. However, instead of returning the data selected by the user directly to the user’s browser, the resource returns a web address for the meta-server to retrieve the data. From the user’s perspective, the interaction is entirely familiar, being exactly the same as when they access the resource directly from their browser. However, when the resulting data are returned at the end of the user’s interaction, they appear as native Galaxy datasets, persisted in the user’s current workspace. Implementing the Galaxy protocol is a relatively simple task for both the Galaxy developers and the data resource programmers, compared to a typical web-services interface. The BioMart and the UCSC genomic repositories currently offer transparent integration with Galaxy, and their implementation experience suggests that the simplicity of the Galaxy interface will allow other major data providers to offer Galaxy style interfaces with relatively little programming effort.

Designing for sustainability

Translational genomic research is a broad church, so it seems unlikely that any individual or even any single research institution could ever design, create and maintain a suitable application and all of the associated data resources and software applications, no matter how wise and well resourced. A collaborative, professionally and centrally managed open-source model, using modern software engineering practices, is a practical and proven approach, and is probably the only mechanism likely to garner sufficient “mind share” amongst the many talented designers, biologists, developers and software engineers needed to create, extend, manage, and support such a large scale undertaking, in the absence of substantial commercial resources.

Documentation will always be a crucial issue in helping users to understand the wide variety of external tools and resources exposed through integrative frameworks like Galaxy. Users will vary in their preferences, but making tutorial demonstrations of illustrative, use-case analyses, and delivering these as needed, as multimedia screencasts, is proving to be an efficient and popular method for spreading knowledge about the meta-application. Written documentation is always valued, but “see one, do one, teach one” is the approach of our documentation and dissemination effort for Galaxy.

Implementation lessons and features

The language chosen to implement the first version of Galaxy was Perl. As the complexity of the code grew, it became increasingly and painfully apparent that this was not a sustainable implementation language for a complex meta-application, being developed by a large and geographically dispersed team. The second and current implementation is in Python[9], and this choice has proven to be scalable and robust. Note that we do not claim that Python is necessarily the best or only possible choice for implementation, but progress to date suggests that it fits our needs very well.

Python is an agile, interpreted language, allowing complex applications to be developed relatively quickly compared to compiled languages, potentially at the cost of limited scalability. In practice, Galaxy performance has proven to be highly scalable, partially because Python itself is no slouch on modern hardware, and partially because the meta-server has been designed to avoid computationally intensive tasks.
Instead, Galaxy has been designed to offload the work of executing tools to more appropriate and scalable infrastructure if available, allowing Galaxy framework developers to focus on developing efficient data structures, low impedance interface specification methods, and by providing transparent user interfaces, freeing more time for the developers of the computationally intensive applications to focus on efficiency. When started using the default distributed configuration file, Galaxy fires up an inbuilt web server, an embedded relational database backend, and runs all user initiated computational tasks as threads on the primary machine. By changing a few lines in the default configuration file, Galaxy can use a remote web-server front end, a remote PostgresSQL or MySQL relational backend, and user-initiated jobs can be offloaded through a simple interface to a computational cluster.

Galaxy’s design is deliberately minimalist. For example, it avoids the complexities associated with web-services, but is compatible with all current and future web services. This paradox arises because Galaxy focuses on facilitating the integration of executable applications as tools, and a wrapped application can be written to negotiate with a remote web service, without any changes to Galaxy itself. This loosely-coupled design contrasts starkly with native web-services-centric meta-server designs (see below), because Galaxy pushes all dependence on the intricate specifics of individual web services interfaces out of the meta-server framework, to individual external executable applications, arguably a far better place for tight coupling, insulating the Galaxy code-base from all the associated truculent complexities.

Software engineering issues

Multiple and potentially concurrent collaborating developers, code versioning and application release cycles are managed using a Subversion repository, integrated with the Trac (http://trac.edgewall.org/) software project management application. Source code is freely available for anonymous checkouts, and a daily “buildbot” process runs a comprehensive suite of unit and regression tests that are specified in Galaxy source code and XML tool interface documents, reporting any errors to the developer mailing list. Galaxy provides comprehensive, automated operational exception error reporting, so if a user encounters an error during the operation of any tool, a detailed description is logged for developer attention, and the user is asked to provide any additional potentially useful information by completing a typical Galaxy interface web form, to be emailed to the developers for rapid response. As a result, software errors are quickly found and fixed, and the main Galaxy instance is proving to be an extremely robust, scalable and reliable resource despite the relatively rapid pace of both development and steady service load growth.

Other meta-applications and application frameworks

Galaxy is competing in a vigorous open-source meta-application space. Many of the ideas implemented in Galaxy are also available in other open-source software projects. For example, Galaxy builds on many design and architectural features, including the XML interface descriptor for executables, in Pise.[10] Taverna[11, 12] is a popular Java framework for constructing custom workflows using local applications and remote web services. The closely integrated BioMoby[13, 14] project provides a powerful, open-source Java based framework integrating remote genomic web-services into complex, publishable workflows with consistent semantics across heterogenous resources. Gene Pattern[15] offers a workspace model currently focused on micro-array expression and proteomic analysis.

There are a very large number of competing commercial products, many derived from the generic business application framework and workflow space, including specialized genomic packages, but none of these are available as free services and none have such transparent integration with major annotation sources. For example, there are products from Rosetta Bioinformatics, and from Inforsense, but these have substantial license fees, and require expert customization for new research tasks.

Proof of the pudding

As a proof-of-concept, a single public Galaxy instance is supporting hundreds of researchers and running thousands of jobs every day (http://g2.trac.bx.psu.edu). All Galaxy tools present a uniform web-based interface to the user, and the outputs from one tool can provide the input for another, allowing complex data workflows to be quickly constructed. The interface adds flexible recording of all research activity, and integrated annotation services for intermediate outputs.

Major genomic data sources are integrated into Galaxy in a transparent manner, so data to be easily imported into a workspace[16]. Many Galaxy tools produce results that can be viewed in full genomic...
context as UCSC custom tracks or genome graphs, with a single mouse click. Galaxy requires minimal end user training and supports reproducible, verifiable research, because analysis steps and intermediate datasets are persisted in each user’s private Galaxy history view, from where they can easily be shared with other users if desired to facilitate reproducible and transparent research.

Conclusion

Currently, there are many competitive technical idioms to choose from for building individual computational tools. Unfortunately, this rich virtual variety creates practical barriers to translational research, through complex technical dependencies, and wildly variable data and command syntax, between analysis packages. Each successful investigator has overcome all these barriers for themselves. This roadblock is unlikely to ever be fully addressed by any single monolithic application, because software developers are unlikely to ever agree on one single language or framework. Galaxy neatly sidesteps this problem by offloading all computation, making it entirely indifferent to all implementation details beyond the command line syntax and output requirements, of applications it incorporates. Every step and all intermediate outputs in a research project performed using Galaxy is recorded in convenient, shareable persistent workspaces, and Galaxy exposes familiar, native interfaces to popular data resources. Best-of-breed third party applications, data and annotation resources can quickly be added to extend the functionality provided through the consistent Galaxy interface. Finally, a Galaxy account is available to any investigator, requiring only an Internet capable web browser, lowering the “price of entry”, effectively helping to commoditize transparent, reproducible, translational genomic research.

Supported by: NIH Grants 5R01HG003646-02, U01 HL065899-05, 5U54LM008748-02 and NSF Grant DBI-0543285

References

[1] National Center for Biotechnology Information. GenBank statistics. [Web page] 2004 [cited 2007 September 2]; Available from: http://www.ncbi.nlm.nih.gov/Genbank/genbankstats.html
[2] HapMap Consortium. HapMap (http://hapmap.org/). 2004.
[3] National Center for Biotechnology Information. dbGaP. 2007 [cited 2007 September 2]; Available from: http://www.ncbi.nlm.nih.gov/sites/entrez?db=gap
[4] Giardine B, Riemer C, Hardison RC, Burhans R, Nekrutenko A, Galaxy: a platform for interactive large-scale genome analysis. Genome Res. 2005 Oct;15(10):1451-5.
[5] Galaxy Team. Galaxy, http://main.g2.bx.psu.edu/, Penn State 2007.
[6] Anonymous. "Borg". [cited 2008 January 18]; Available from: http://memory-alpha.org/en/wiki/Borg
[7] Nekrutenko A. Galaxy installation demonstration, http://screencast.g2.bx.psu.edu/GR_Screencast_18.mov. 2007.
[8] Kent WJ, Sugnet CW, Furey TS, Roskin KM, Pringle TH, Zahler AM, et al. The Human Genome Browser at UCSC. Genome Research. 2002;12:996-1006.
[9] van Rossum G. Python. http://python.org, version 2.4.2 ed. MA: Python Software Foundation 2006.
[10] Letondal C. A Web interface generator for molecular biology programs in Unix. Bioinformatics. 2001;17(1):73-82.
[11] Hull D, Wolstencroft K, Stevens R, Goble C, Pocock MR, Li P, et al. Taverna: a tool for building and running workflows of services. Nucleic Acids Res. 2006;34(Web Server Issue):W729-W32.
[12] MyGrid Project. Taverna. http://taverna.sourceforge.net/ 2007.
[13] BioMoby. http://biomoby.org 2007.
[14] Wilkinson M, Links M. BioMOBY: an open-source biological web services proposal. Briefings in Bioinformatics. 2002;3(4):331-41.
[15] Reich M, Liefeld T, Gould J, Lerner J, Tamayo P, Mesirov J. GenePattern 2.0. Nat Genet. 2006;38(5):500-1.
[16] Nekrutenko A. Galaxy demonstration - identify all human dnase hypersensitive sites upstream of a gene. http://screencast.g2.bx.psu.edu/MainUseExample.mov. 2007.