GetGeo: An Integrated Data Acquisition Tool for Galaxy

Aryeh Hillman

December 16, 2012

Abstract

Legacy bioinformatic databases, such as the National Center for Biotechnology's Gene Expression Omnibus (GEO) [4] database contain valuable information about human biology. Despite the plethora of data available on GEO, however, data is often poorly organized and often lacks useful meta-information, making it difficult to use the existing data with modern computational biological algorithms. To address this problem, a tool was used to fetch curated information from GEO for the Galaxy tool [6][2][5], a web-based platform for biomedical research, thereby allowing hundreds of powerful, existing algorithms to make use of data from GEO.

1 Background

During the late nineteen-nineties, due to the development of commercially available, mass-produced RNA chips, large amounts of information regarding RNA expression were being produced across the world – even before the completion of the human genome project. Yet despite such incredible amounts of information and the already relative-popularity of the internet, much of this data was not easily available to the public and was not organized in a uniform fashion. While some organizations such as the Brown Lab at Stanford were pioneers in putting data online, other labs were slow to follow, despite receiving funding from public agencies such as the National Institute of Health (NIH)[7].

Needless to say, some individuals spoke up about this issue in two important papers in Nature, one mentioning that "universal standards to make the data more suitable for comparative analysis and for interoperability with other information resources have yet to emerge;"[7] the other mentioning important, still largely unaddressed pitfalls regarding consistency

between experiments, but also mentioning the need for "structured information storage."[3] Noting this increasing pressure from both the public and the development of databases elsewhere, such as one developed by the European Bioinformatics Institute (EBI), the NIH began developing and implementing GEO, which it released in 2006[4]. GEO met many of the needs of the public at the time, providing a centralized location for bioinformatic datasets, especially those related to RNA expression.

While GEO has been a valuable resource for several years, due in part to the ease with which scientists can submit datasets which vary in format, it has also suffered as it neither imposes nor validates any sort of structure on data submitted, be it actual quantitative data or descriptions of samples which are analyzed.

2 The "Get GEO" Tool

Despite certain pitfalls with GEO, we were interested in creating a tool that would help retrieve datasets in GEO and put them in a consistent format, which would allow modern algorithms to analyze large numbers of datasets.

Fortunately, we discovered that some other individuals had worked hard to help bring GEO under the reigns of certain degrees of consistency. For example, the NIH relatively recently developed the GEO2R tool, which allows an arbitrary dataset to be pulled into R, which makes use of the Biobase and GEOquery R tools, code which retrieves data from GEO and places them into consistent data structures.

While good mechanisms to retrieve quantitative information from GEO exist, we noticed that there were not good methods to retrieve qualitative information about datasets[1]. Qualitative information about a given dataset is as important as the data itself. Borrowing from the practice of machine learning, the quantitative data are like features, whereas qualitative data are like labels. For example, if a scientist wished to perform research on cancer, for example, with the hypothesis that certain RNA sequences were more common, the scientist would need to know which datasets had information about cancer and which datasets had information about normal cells – in this example, the qualitative label is "cancerous" or "non-cancerous" and the quantitative data are the datasets, which describe the relative abundance of RNA sequences.

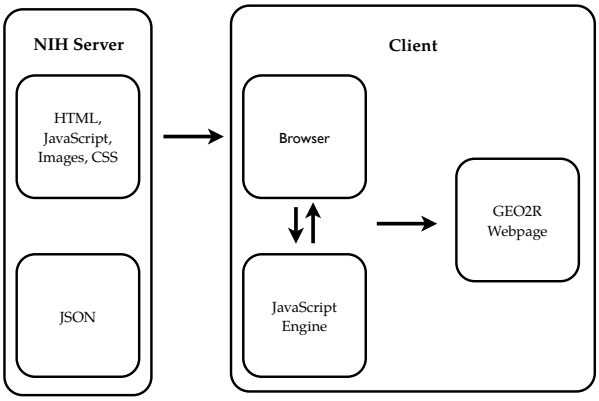

Figure 1: GEO2R Page Rendering

2.1 Retrieving Descriptive Information

Our first problem was that qualitative information was not available in an easily downloadable format. We noticed that useful descriptive information was available in the form of a JQuery table on the GEO2R website. This data table was populated by JavaScript, however, thus HTML code could not simply be parsed for the underlying data.

One option we considered was evaluating whether it might be possible to evaluate Javascript on an HTML document offline and outside of the context of a full-blown browser. While this approach seems somewhat inefficient, it seemed worthwhile, if it meant the retrieval of data into a structured format. This approach, however, was not easily realized. While many JavaScript engines exist apart from browsers (e.g. Google's V8 and WebKit's JavaScript engine, and PhantomJS), it seemed that successfully executing JavaScript on HTML could require a lot of time spent learning about the Document Object Model (DOM) and how command line interface JavaScript engines work.

After spending time with the GEO2R page source code, it became clear that the descriptive tables were being populated by downloading and parsing JSON files from the NIH. [8]

Retrieving the underlying JSON files wasn't too challenging, as GEO uses several server-side scripts to generate them which are accessible through simple `HTTP GET` requests. Requests can be made to discover useful information such as which platforms are associated with a given accession and can be used to get complete descriptive information about a given dataset, such as information about phenotype and the lab from which the data was

received.

```
http://www.ncbi.nlm.nih.gov/geo/tools/geometa.cgi?
  &series=GSE21032&view=samples&platform=GPL10264
```

Figure 2: Example URL to generate JSON descriptive information

Resulting data structures, in particular, those with information about all samples described by a particular accession and platform, are not always uniform in structure. These data structures are often quite complex and are comprised of seemingly arbitrarily deep nests of dictionaries and lists. Parsing these data structures was a bit difficult, because what we really care about are only keys and associated values represented in the form of ASCII comma-separated value files. To traverse the resulting JSON data structure, an algorithm similar to the one below was developed to flatten the data structure[9]:

```
def flatten(l):
    out = []
    if isinstance(l, (list, tuple)):
        for item in l:
            out.extend(flatten(item))
    elif isinstance(l, (dict)):
        for dictkey in l.keys():
            out.extend(flatten(l[dictkey]))
    elif isinstance(l, (str, int, unicode)):
        out.append(l)
    return out
```

Figure 3: Flattening Algorithm

2.2 Retrieving Quantitative Data

Fetching the quantitative information associated with each platform and accession was fairly straightforward. This task was delegated to R, as at the moment, there are some very existing tools, namely `Biobase` and `GeoQuery`, which are maintained by the Bioconductor Project. There are effectively two things which these underlying tools are doing. First, given an accession and a platform, raw data is retrieved from GEO. These data are "raw" because they use a proprietary namespace defined by an RNA chip manufacturer

to refer to RNA's that are found on a given RNA chip. Second, given the raw data, the proprietary namespace is mapped to a general namespace that refer to a particular RNA sequence, regardless of the vintage or the manufacturer of a given RNA chip. This process could have been delegated to hand-written Python code, but the mapping process is quite cumbersome and can require both parsing and downloading definition files – a process that can be prone to error and that has been addressed well by Bioconductor.

Following a check on the validity of a given accession and platform, a short script is piped to R over `STDIN`, the result of which gathers the quantitative data of interest.

3 Integration with Galaxy

3.1 Introduction

Galaxy, a "platform for interactive large-scale genome analysis" was introduced in late 2005[6]. Galaxy is a web-based tool, which allows for the storage of data and the execution of algorithms using a web-based interface. The tool was developed in response to powerful, existing data retrieval and visualization tools, such as UCSC's Genome Browser and NCBI MapViewer applications. While these tools were very useful for biologists, they were relatively limited as computational tools for experiments, as they were unable to perform certain kinds of experiments on data. In the past, this meant such analyses as "what is the distribution and conservation of a gene among several genomes." Today, such questions that might be within the scope of Galaxy's power, should the tools be developed, might include "what genes are expressed during stage three breast cancer?"

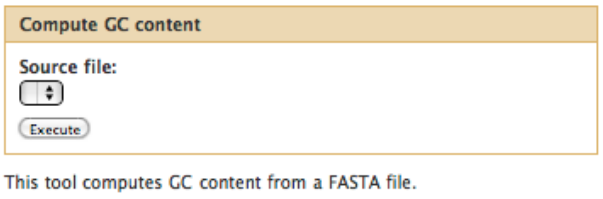

Figure 4: Adapted tool `gcContent.pl` ⟨`FASTA file`⟩ ⟨`output file`⟩[10]

While UCSC's genome browser has grown in capability over the years, its development is ad hoc, as UCSC's software must rely on relatively simple frameworks, such as PHP. By contrast, Galaxy provides facilities for easily

adapting existing command-line software through the use of XML wrappers. Taken together, a the use of a group of tools in Galaxy can be called a "workflow," which might allow a dataset in one format to be converted to various other formats and analyzed by several different algorithms. Using traditional techniques, these experimental methodologies would have been very difficult to execute, due to varying configurations on scientist's computers and would have required writing a lot of code from scratch to address particular problems, such as the conversion of data sets from a binary format to an ASCII format, which can be very time-consuming and prone to error.

3.2 GetGeo Galaxy Adaptation

Many existing experimental command line tools operate like tools in the base UNIX package, in that they take parameters before running and produce an output. These sorts of applications are easily adapted for usage in GEO. This is because Galaxy's XML wrapper implementation creates a simple web-interface that allows specification of objects including drop-down lists and text-boxes which can automatically be populated with files that exist on the galaxy platform.

Even certain kinds of interactive tools are adaptable for Galaxy. For example, one could write a simple algorithm to interact with an interactive app, obviating the need for interaction except for entering information in the web-interface before running the application.

To some degree, Galaxy even provides facilities for a certain degree of interactivity with applications, when it is more convenient for the user. For example, Galaxy provides facilities for sanitizing and otherwise checking input to tools by delegating input to intermediate scripts. Also, the web-form interface for a tool can even be populated with dynamic information using a specified script.

Yet one use-case that Galaxy does not appear to cover is interactivity wherein a user enters something into the web-interface, a script is called, and the user is prompted with a new set of options. In the case of the GetGeo tool, this user interface pattern would have been very useful.[11]

The reason is that each accession on GEO might be associated with a variable number of platforms. It would have been very useful to have been able to fetch a given accession and display some information about it and its associated platforms, giving a user the chance to validate their choice of accession and the ability to choose between various platforms. Instead, given the current limitations of GEO, the Galaxy interface for the GetGeo

tool had to be coded to accept both the accession and platform parameters up front, requiring the user to reference the GEO tool itself. While not terrible, some other data-acquisition tools on GEO such as the interface to UCSC's Genome Browser are more interactive in the regard discussed.

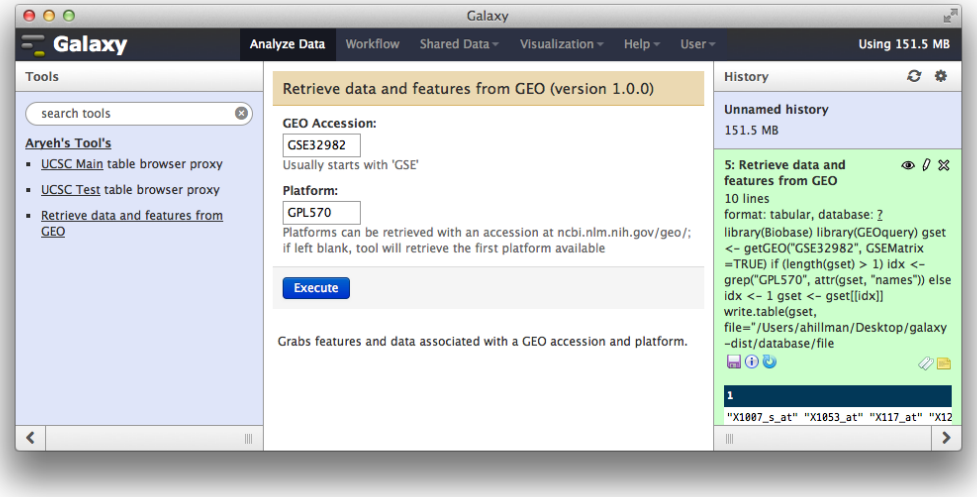

Figure 5: GetGeo's Galaxy Web Interface

To use the tool, a user navigates to the tool using the "Tools" pane on the left-hand side of the window. The tool is clicked on and a GEO accession and platform is entered. If a platform is not entered, the tool asks GEO for a list of platforms associated with the accession, and automatically downloads the first platform returned (this feature is most useful if the user knows that there is only one platform associated with a given accession). After a few minutes to an hour, the files of interest (a file which describes each sample associated with a platform and a file with RNA expression information). The process can take quite some time, especially for large files, as it can be time consuming to retrieve and parse definition files associated with a given RNA chip.

4 Future Work

Ideally, a tool more similar to UCSC's Genome Browser would be ideal, featuring a higher degree of user interactivity as discussed earlier. We at-

tempted to develop a tool like UCSC's, but two key challenges were encountered, which could not be resolved in the time allotted for this project. Both challenges stem from the fact that the UCSC Browser Tool is hosted off-site, on UCSC servers. Namely:

1. To enable a higher degree of interactivity, an alternative rendering system apart from Galaxy's XML wrapper would have to be used. There are many options for such an implementation, such as the use of of a web language like PHP or a framework such as Python's Django. This would have been possible, but would have required persistent off-site hosting. Moreover, such an implementation might not be well-received by the NCBI, as it would represent a free-standing tool that extends functionality of GEO, but still relies on GEO resources. A more attractive solution would would involve writing the web tool using Galaxy's web application rendering facilities. We considered this option, as it appears that an earlier version of Galaxy effectively used this second strategy to implement proxy-like access to the UCSC Genome Browser. While this approach would have probably been tenable, we were unable to consummate this approach, as the framework which renders Galaxy's backend did not appear to be well documented at this point.

2. Tools which do not rely upon Galaxy's XML wrapper require forwarding files to the Galaxy server, a process which is normally handled by Galaxy by providing special files to a given tool. This process is fairly well-documented as several data-acquisition tools implement this technique. In particular, a special URL, call it "URL X" is handed to a data repository of interest. When the user selects data of interest from the repository, the repository's servers pass the data back to Galaxy using "URL X."

Another tack would be to improve the Galaxy XML system. Improving this system would doubtless be useful for many users.

Contingent upon the utility of the GetGeo tool for users of Galaxy, it would also be convenient to cache RNA Chip definition files to improve the efficiency of the tool.

5 Conclusions

Galaxy is an exciting tool which will allow experimental biomedical research to be conducted. Among its strengths are its ability to work with and retrieve a plethora of existing bioinformatics data which is available across

many different sources; and its ability to host and share experimental algorithms.

From the standpoint of the development of a data acquisition tool, Galaxy does have some weaknesses. A higher degree of interactivity from the XML wrapping system would be highly desirable. Moreover, it would be excellent were it possible to have some sort of system that could arbitrarily wrap-around existing data-sources.

In developing this tool, we became more familiar with Galaxy and the GEO database. We hope that this tool may be useful for users of Galaxy in the near future.

6 Acknowledgements

I would like to thank Robert Baertsch for help finding a useful project to work on and for his valuable initial explanations of GEO and Galaxy. I would like to thank Luca de Alfaro for sponsoring my independent studies and for suggestions and inspiration about how to approach software engineering, which I learned in his graduate course last year. Finally, and most importantly, I would like to thank Theodore Goldstein for general suggestions about how to approach software engineering from the standpoint of both theory and design and for guiding me towards work in bioinformatics.

References

[1] Tanya Barrett1, Stephen E. Wilhite, Pierre Ledoux, Carlos Evangelista, Irene F. Kim1, Maxim Tomashevsky, Kimberly A. Marshall, Katherine H. Phillippy, Patti M. Sherman, Michelle Holko, Andrey Yefanov, Hyeseung Lee, Naigong Zhang, Cynthia L. Robertson, Nadezhda Serova, Sean Davis, and Alexandra Soboleva. Ncbi geo: archive for functional genomics data sets - update. *Nucleic Acids Research*, 2012.

[2] D Blankenberg, G Von Kuster, N Coraor, G Ananda, R Lazarus, M Mangan, A Nekrutenko, and J Taylor. *Current Protocols in Molecular Biology*. January 2010. Chapter 19:Unit 19.10.1-21.

[3] Alvis Brazma, Alan Robinson, Graham Cameron, and Michael Ashburner. One-stop shop for microarray data. *Nature*, 403(699-700), February 2000.

[4] Ron Edgar, Michael Domrachev, and Alex E. Lash. Gene expression omnibus: Ncbi gene expression and hybridization array data repository. *Nucleic Acids Research*, 30(1), 2002.

[5] B Giardine, C Riemer, RC Hardison, R Burhans, L Elnitski, P Shah, Y Zhang, D Blankenberg, I Albert, J Taylor, W Miller, WJ Kent, and A Nekrutenko.

[6] J Goecks, A Nekrutenko, J Taylor, and The Galaxy Team. Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. *Genome Biology*, 11(8), August 2010.

[7] Douglas E. Bassett Jr., Michael B. Eisen, and Mark S. Boguski. Gene expression informatics - it's all in your mine. *Nature Genetics*, 21(1), 1999.

[8] National Center for Biotechnology Information. GEO2R, 2012. `http://www.ncbi.nlm.nih.gov/geo/geo2r/`.

[9] StackExchange. Flattening a list of dicts of lists of dicts (etc) of unknown depth in python (nightmarish json structure), 2011. `http://stackoverflow.com/questions/8477550/`.

[10] Galaxy Team. Adding tools to galaxy, 2012. `http://wiki.galaxyproject.org/Admin/Tools/Add%20Tool%20Tutorial`.

[11] Galaxy Team. Tool configuration syntax, 2012. `http://wiki.galaxyproject.org/Admin/Tools/ToolConfigSyntax`.