BioMashups: the new world of exploratory bioinformatics?
Jiro Sumitomo, James M. Hogan and Paul Roe
Microsoft QUT eResearch Centre
Queensland University of Technology
GPO Box 2434, Brisbane, QLD, 4001
AUSTRALIA
j.hogan@qut.edu.au

Bioinformaticians have a large range of tools and services at their disposal, each of which is useful alone, but more powerful when integrated with others to perform a more complex task. Routine workflows are a critical aspect of modern science, but the programming required to realize them may act as a barrier to free exploration. Web mashups provide one solution to this problem, rapidly combining data and services from multiple sources into a single facility through a web based application.

Mashups are growing in popularity in the life sciences, but like their mainstream equivalents, BioMashups are generally written from scratch, and independently hosted. Mashups are small, focused applications with limited scope, their appeal dependent on a combination of services critical to the solution of a particular problem. The specificity of bioinformatic applications and the absence of suitable component libraries may act as barriers to mashup sharing and broader adoption. Moreover, many active biologists lack the programming skills necessary to integrate novel mashups, and greater support is necessary.

Recently, a number of sophisticated frameworks for developing and hosting mashups have emerged, each markedly simplifying the mashup creation and sharing process, although these environments vary in their focus. Notable among these initiatives are Microsoft Popfly1, Google Mashups2, Yahoo Pipes3, Intel MashMaker4 and Lotus Mashups5. We believe that these frameworks may in time address some of the concerns identified above, making novel mashups accessible to less technical users. In particular, given a broad range of useful component blocks to compose, it is likely that Popfly will prove a very convenient environment for exploratory bioinformatics.

To create a mashup, the author creates a wrapper around the separate tools and services to be composed. Each wrapper has an interface known to the author, and so wrapped tools and services may interact with each other. Frameworks promote a standard means of building these wrappers, supporting a common interface between different service components. Over time, adoption of suitable frameworks within the scientific community may well lead to a community based component library of some complexity. However, this adoption is dependent on a broader awareness of the capability of the mashup

1 http://www.popfly.com
2 http://editor.googlemashups.com
3 http://pipes.yahoo.com
4 http://mashmaker.intel.com
5 http://www.ibm.com/software/lotus/products/mashups
approach, and perhaps upon a number of groups seeding the environment with components of sufficient utility.

In this tech report, we consider these issues in some detail and offer an example of the emerging mashup capability using the Popfly environment from Microsoft. We have created three demonstration mashups. The first is a simple lookup, mapping Uniprot\(^6\) Protein Accession Ids to their GenBank\(^7\) equivalents. The second mashup retrieves articles from the Pubmed\(^8\) database that reference a specified GenBank Accession Id. This mashup is immediately useful to a researcher who has a known gene of interest and wants to find all known information about the gene; it also forms a very useful component of a more elaborate annotation mashup. The final demo is based on a fairly elaborate undergraduate teaching exercise from QUT, in which the student is given a protein coding nucleotide sequence, and is asked to determine the gene family and to perform an NCBI Entrez search to obtain more information about the gene.

We now examine these mashups in more detail:

**Example Mashups**

Microsoft Popfly is a mashup creation and hosting tool. In Popfly, the basic unit of service composition is known as a block. Blocks can be added to a mashup through drag and drop actions onto a visual mashup composition tool. Data exchanges between blocks are specified by dragging arrows between connectors on blocks.

Like all mashup frameworks, Popfly removes the cost and effort required to host mashups, thereby providing an incentive for bioinformaticians to share applications, even those with a narrow scope. Like Yahoo Pipes and Lotus Mashups, Popfly provides a simple interface to compose blocks to create mashups, making mashup creation accessible to users who are not necessarily programmers. Popfly is unique in that it provides a platform to host and share blocks or components, which users can optionally modify and incorporate into their mashups.

Although the block composition requires no programming, the blocks themselves have to be implemented using javascript, HTML and optionally Silverlight\(^9\). We found block creation to be simple, and well supported using tools such as the Visual Studio plugin, Popfly Explorer. The ability to “rip”, or obtain the source code for shared blocks, not only proved to be an invaluable learning resource, but allowed us to adapt blocks published by Microsoft for use in our mashups. As block logic is written in Javascript, implementing functionality differs little from traditional mashups.

---

\(^6\) http://www.uniprot.org

\(^7\) http://www.ncbi.nlm.nih.gov/Genbank

\(^8\) http://www.ncbi.nlm.nih.gov/pubmed

\(^9\) http://silverlight.net
One notable bioinformatics blog\(^{10}\) expressed disappointment with Popfly due to the lack of built in support for ATOM feeds, a feature provided out of the box in Yahoo Pipes and Google Mashups. While there is some validity in this view, as many mashups are based around RSS/ATOM, the deficiency may be rectified by creating and sharing an appropriate block.

We found the mashup creation process in Google Mashups to be similar to creating an ASP.NET page with specialised controls for working with RSS and ATOM, supplemented by the obligatory map. The outstanding feature of Google Mashups is the ability to persist data on an application wide or user level, which is presently lacking in Popfly. In Popfly, a limited amount of data can be stored at a mashup level to be shared amongst all users.

Google Mashups, however, does not support user created controls, although arbitrary html and script can be inserted into a Google Mashup page. While there is no significant loss of flexibility within the one application, the Popfly block mechanism allows far more convenient sharing and re-use of user implemented functionality.

The primary function of Yahoo Pipes is to enable the manipulation of web based data sources - RSS or ATOM feeds, JSON objects, HTML or Plain Old XML – to produce new feeds through operations such as aggregation and filtering. The resulting Pipe is viewable from the Yahoo Pipes site, and may be embedded within web pages or composed within external mashups. Pipes are constructed by connecting together a set of modules to form a data flow graph, a task performed visually and intuitively through the Yahoo Pipes editor.

However, in keeping with its data manipulation design goals, Yahoo Pipes is not a general purpose mashup composition and hosting framework. Users cannot create their own modules, and are limited in the ways that they can interact with data and control how output is presented. Unlike Popfly or Google Mashups, Yahoo Pipes does not host pages with arbitrary content and script, only pipes. Pipes can be used as part of Popfly or Google Mashup mashups, so Pipes may serve as a companion data source for mashups in other frameworks.

Intel MashMaker is a powerful data integration tool, based on a browser plugin. It enables users to customize exiting web pages with widgets which are able to integrate data from that page, with data from other sources. We would argue that this approach differs from traditional mashups because the focus is on tailoring the web browsing experience to an individual preference, rather than creating new web application to share with others.

We hope to evaluate Lotus Mashups when it becomes available.

**Uniprot - Genbank Id translator**

http://www.popfly.com/users/jsum/Uniprot%20to%20Genbank%20X-ref

---

\(^{10}\) Flags and Lollipops, http://www.ghastlyfop.com/blog/2008/05/disappointed-with-popfly.html
NCBI’s GenBank database is a collection of publicly available annotated nucleotide sequences, including mRNA sequences with coding regions, segments of genomic DNA with a single gene or multiple genes, and ribosomal RNA gene clusters.\(^{11}\)

If part of a GenBank nucleotide sequence encodes a protein, a translation is annotated and assigned a protein accession number. This is noted on the GenBank record and linked to a record for the protein sequence in NCBI’s protein databases. Additional information about these protein sequences may be available from the UniProt (Swiss-Prot) databases, so NCBI’s records provide links to UniProt sequences.

However, the reverse reference does not exist. UniProt entries do not contain references to the corresponding GenBank sequences. It is thus useful for a researcher who has a UniProt accession Id to obtain a GenBank Accession Id to make use of the tools and services provided by the NCBI.

We find the cross references through a service called Bio2Rdf\(^{12}\) which maintains a database of RDF documents on bioinformatics subjects. Any GenBank references are included as triples in the RDF document record of a UniProt protein. We can simply parse this document, which is optionally served in XML format, and extract the GenBank Ids.

As we’re only taking advantage of one service, this is more a wrapper than a true mashup (not including Popfly itself as a service), but it serves as a gentle introduction to the capabilities of Popfly.

This translation tool is implemented as a mashup of four blocks (Figure 1). The first block takes a UniProt accession Id as input, which is entered by the user through a text box. This Id is converted into a URI and fed to the next block, the RDFXmlParser. This block loads an RDF document in XML format, and creates an object collection of all of the triples contained in the document. This collection is sent to the next block in the chain, the Bio2Rdf block. This block understands namespaces and triples as used by the Bio2Rdf service, and extracts the statements of interest to the user; here the GenBank cross references. These statements are extracted and sent to the last block, which simply displays the GenBank Accession IDs to the user.


\(^{12}\) http://bio2rdf.org
Although we used the service exposed by bio2rdf.org, there are other services that provide the same capabilities. For example, UniProt have a service accessible at http://beta.uniprot.org which may be used instead.
Given a Genbank ID, this mashup retrieves articles from the Pubmed database that reference this ID, a workflow plainly of interest to most genomics researchers.

A simple way to find Pubmed references is to access the Bio2RDF document corresponding to the gene of interest. We reuse two blocks from the earlier mashup; thus, only two new blocks are needed.

The first block takes a user input and constructs a URI. This is given to the RDF parser and bio2rdf blocks that were described earlier. Here, instead of extracting GenBank Accession Ids, the bio2rdf block retrieves all PubMed references. These references are uniquely identified by a URI, which is used to retrieve the RDF record for the PubMed article. The bio2rdf block is reused in the mashup, this time to extract the paper abstract, a url link to the PubMed server, and the author list. These results are then fed to the final block, which displays all of the information to the user.

There are other servers that we could have used to implement this mashup, but Bio2RDF is particularly useful, because the record for a Gene ID contains all its aliases or synonyms; such variations abound in the literature. By using the bio2Rdf record for a gene we can obtain all of these synonyms with one lookup instead of introducing an additional pre-processing step.
Figure 3: Article Viewer block composition
At time of publication, this mashup only renders correctly in Internet Explorer; we will extend this to work correctly in Firefox shortly – there is no browser limitation inherent in Popfly.

**SLDM1**
http://www.popfly.com/users/jsum/SLDM1

The final mashup is based on an undergraduate exercise in genomics from QUT. Here, the student is given a protein coding nucleotide sequence, and is asked to determine the gene family, and to perform an NCBI Entrez search to obtain more information about the gene.

To complete the exercise, a student must therefore:
- Perform a Blast (blastx) on the sequence
- Obtain the best hit/hits by inspection of the output page
- Retrieve the GenBank record of the best hit by clicking on the link provided in the blast output page.
- Determine the known region by inspecting the GenBank Record
- Perform an Entrez search on this region
This exercise is perhaps a more realistic test of the utility of the Popfly environment for bioinformatics, and involves many of the more common web-based tools used in the community. An experienced user can manually execute this workflow quite rapidly, but the process is inconvenient due to the need to cut and paste data records of some size and complexity. The mashup eliminates these difficulties, and supports rapid reconfiguration of the process.

We used seven blocks for this final mashup. The first block is a user interface which enables users to select blast options and parameters. This block is really a limited version of the interface provided by NCBI. The second block invokes NCBI’s blast service with the user specified data and parameters. When the blast is complete, the output is sent to the analyzer block, which extracts the best hit or hits (depending on how the user configured the block). The hits are converted to URIs, and information about the hits is obtained from the bio2rdf service. In this mashup, the bio2rdf block is used to extract and output known regions, which are used as inputs to an NCBI Entrez search. The results of this search are then presented to the user.

Figure 5: Self Directed Learning Module 1 (SDLM1) block composition
Conclusion

There are now an enormous number of data resources and services available to bioinformaticians\(^\text{13}\) and an almost infinite number of ways in which they can be usefully composed. So when seeking to compose a set of services to solve a given problem, it is

unlikely that a bioinformatician will find a pre-existing application addressing this particular need. Further, if by chance such an application has been made publicly available and is found by the bioinformatician, the heterogeneity of computing languages and platforms may make that application unusable. Often the bioinformatician has no option but to write a new application from scratch. These issues of application specificity and the lack of a common interface in turn provides little incentive for the bioinformatician to make their new application publicly accessible, particularly if there is an additional effort and cost involved in arranging hosting.

The benefit of Mashup frameworks such as Popfly in bioinformatics is that they promote a culture of sharing and reuse, providing a convenient environment for creating applications and eliminating the hosting barrier. The particular nature of Popfly blocks allows a generalization of the mashup approach and enables partial use of an existing solution. Traditional mashups, and other applications that compose services are an all or nothing proposition. If a user wants to compose services A and B, but an existing mashup composes services A and C, then the existing mashup is wholly unsuitable. The block compositional nature of Popfly means that the user can re-use part of the existing mashup that wraps service A, and needs only to provide a block that wraps service B. If there is an existing mashup that somehow makes use of service B, then it too can potentially be reused.

While each of the mashup frameworks may support lightweight bioinformatic exploration, the Popfly block composition model provides some significant incentives to implement solutions within the mashup framework. The framework provides a library of reusable blocks, making service composition applications easier to write. Often, only minor changes to existing blocks will be required, and no new code will be required at all. In common with the other offerings, the framework offers a uniform platform, so issues arising from heterogeneous languages, operating systems and environments are masked. There is a standard way that blocks communicate with each other, so interoperability of blocks is straightforward, and block composition into mashups is simple and can easily be accomplished by non programmers. We would hope that with enough participation, the potential for mashup frameworks to simplify bioinformatics application development and foster their reuse will be realised.