Vol. 1 No. 3
March 2008

Bioinformatics
Organization

Unveiling Pipet (part two)

J.W. Bizzaro

We've been revealing the unique aspects of the Pipet collaborative research environment in this multipart article. This month, we'll delve into the declaration and definition of Pipet cells, a generic container for graphical user interface (GUI) and command-line interface (CLI) components.

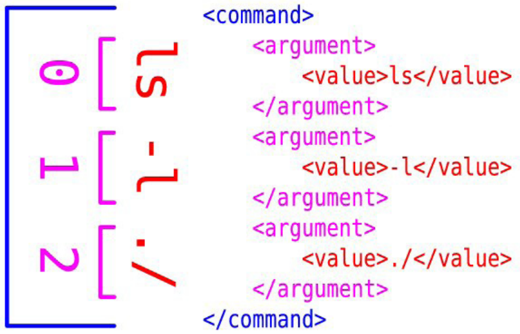
Pipet can work something like a wiki system in which "templates" have been replaced by "cells." So, interacting with cells can be easy. For instance, the software behind Wikipedia.org (MediaWiki) uses the following directive to add the contents of one page (the template) to another:

```
{{templatename|parname1=parvalue1|parname2=parvalue2|...}}
```

And it's the same in Pipet: the directive calls a cell and inserts the cell's contents in its place. Parameters can also be used, separated by vertical bars (|). This works a lot like adding a function call to a print statement in a common programming language. As an example, suppose we have a cell named "List", previously created for the ubiquitous Unix "ls" command. We could then enter the following in a new, empty cell:

```
Here are the contents of the FTP directory:
{{List|Directory=/ftp/pub}}
```

Here, a call is made to the cell, passing along the parameter "Directory=/ftp/pub". By default, this cell contains an XML description of the command "ls -l ./", which contains three arguments: (0) ls, (1) -l, and (2) ./ (see Figure 1).



```
<command>
    <argument>
        <value>ls</value>
    </argument>
    <argument>
        <value>-l</value>
    </argument>
    <argument>
        <value>./</value>
    </argument>
</command>
```

Figure 1: The Unix command for listing a directory, as described by the Pipet Cell Markup Language (sans attributes). The command characters (in red) are oriented vertically to match their respective XML elements.

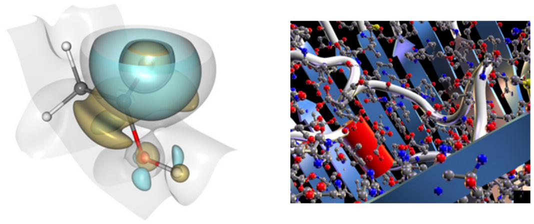
For the uninitiated, the first argument in the command—programmatically numbered 0 and commonly considered the command itself—is a program that lists the contents of a directory. The Linux manual entry actually says, "List

Continued on page 2.

Project spotlight: Molekel

Molekel is an open-source (GPL) multiplatform molecular visualization program being developed at the Swiss National Supercomputing Centre (CSCS). Some of the features available inlude: Multiplatform: Mac OS X, Windows, Linux

- Visualization of residues (ribbon or schematic)
- Complete control over the generation of molecular surfaces (bounding box and resolution)
- Visualization of the following surfaces:
    - Orbitals
    - Iso-surface from density matrix
    - Iso-surface from Gaussian cube grid data
    - SAS
    - SES
    - Van der Waals
- Animation of molecular surfaces
- Export to PostScript



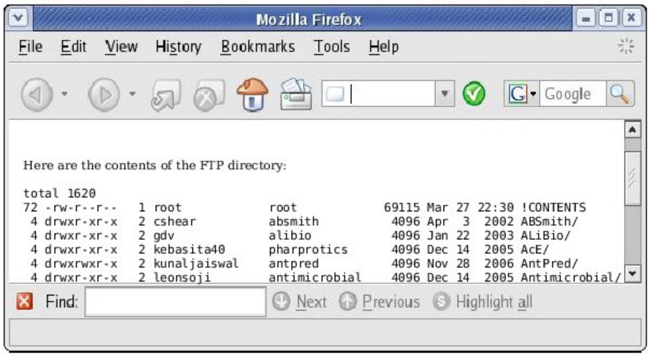
- Export animation
- Plane widget to visualize a scalar field: the plane can be freely moved in 3d space and the points on the plane surface will be colored according to the value of the scalar field: a cursor can be moved on the plane surface to show the exact value of the field at a specific point in space.

Molekel has been hosted at Bioinformatics.Org since 2006 and is available at http://bifx.org/molekel/.

Continued from page 1.

information about the [specified] files (the current directory by default).” So, skipping over to argument number 2, `./` is our specified directory, the current directory (since this is the default, it is in fact redundant, but let’s keep it in for this example).

In-between the program name and the specified directory is an option, `-l`. This option specifies that the output is to be printed in “long format,” with additional details. Figure 2 is an example of what this would look like from the Bioinformatics.Org FTP directory. Had other options been used instead of, or in addition to, `-l`, the output would be different. (On a Unix/Linux computer, type `man ls` for more information on this command and its options.)



Here are the contents of the FTP directory:

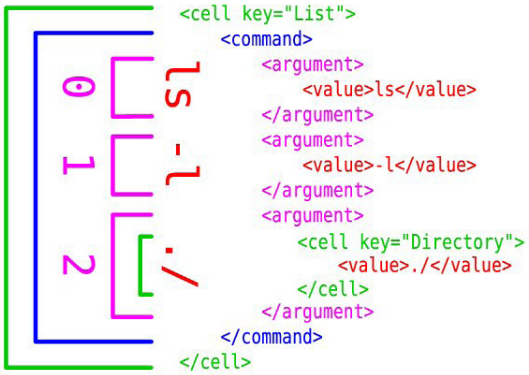
```
total 1620
72 -rw-r--r--   1 root           root          69115 Mar 27 22:30 !CONTENTS
 4 drwxr-xr-x   2 cshear         absmith        4096 Apr  3  2002 ABSmith/
 4 drwxr-xr-x   2 gdv            alibio         4096 Jan 22  2003 ALiBio/
 4 drwxr-xr-x   2 kebasita40     pharprotics    4096 Dec 14  2005 AcE/
 4 drwxrwxr-x   2 kunaljaiswal   antpred        4096 Nov 28  2006 AntPred/
 4 drwxr-xr-x   2 leonsoji       antimicrobial  4096 Dec 14  2005 Antimicrobial/
```

Figure 2: Wiki-formatted text, including the output returned by a hypothetical “`List`” cell.

In a similar fashion, Pipet Cell Markup Language contains “`<command>`” elements, which contain a number of “`<argument>`” elements, and they in turn contain text (or character data) denoted by “`<value>`” elements. Furthermore, in Figure 3, you’ll see that all of the elements can be contained within a single “`<cell>`” element (where `key=”List”`). Recall that a cell is a generic container.

Continued on page 3.

Continued from page 2.



```
<cell key="List">
  <command>
    <argument>
      <value>ls</value>
    </argument>
    <argument>
      <value>-l</value>
    </argument>
    <argument>
      <cell key="Directory">
        <value>./</value>
      </cell>
    </argument>
  </command>
</cell>
```

Figure 3: The Unix command for listing a directory, including two cells.

Being a container, a cell also has a key-value relationship with whatever character data it contains, at any level of nesting. Within Pipet, our hypothetical "`List`" cell becomes a variable that is declared via XML. Its default definition (without the execution of the command) is the sum of the character data in the `<value>` elements, which is "`ls -l ./`". In fact, our "`Directory=/ftp/pub`" parameter is also represented by a cell (see Figure 3), and passing this parameter along to the cell replaces the default definition.

If we were to continue to add cells with additional commands to the XML, we would form a script where one or more command arguments can appear as options to be set by a Web user.

In next month's installment, we'll look into cell compilation and its affect on the GUI. If you have any questions or would like to get involved in Pipet, please contact jeff@bioinformatics.org.

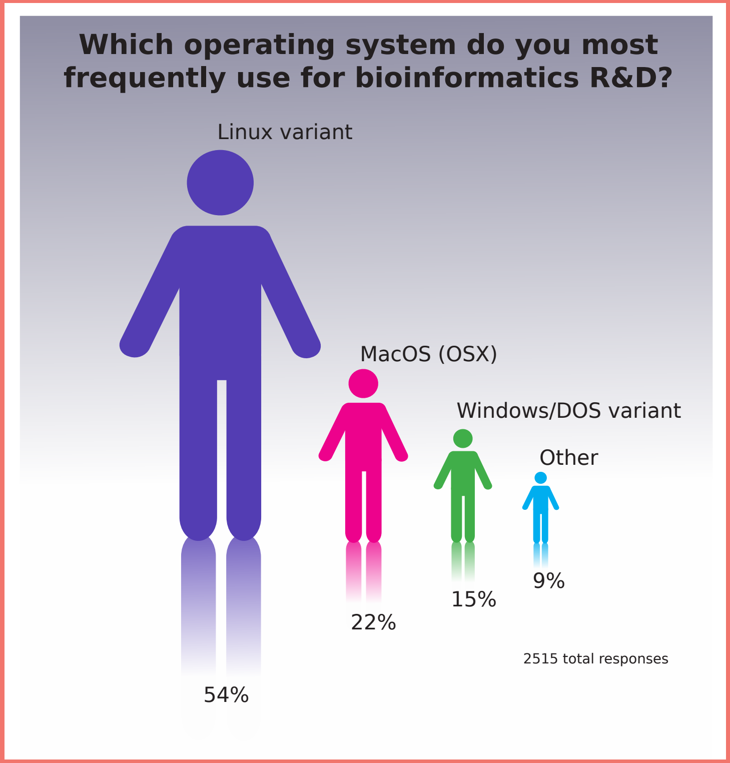
Job search

Looking for a job in the bioinformatics field?

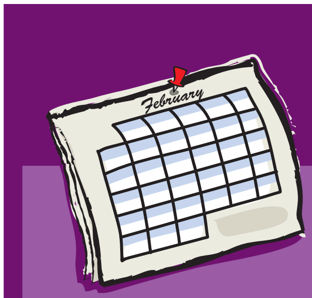
The USDA, Agricultural Research Service, Bovine Functional Genomics Laboratory in Beltsville, Maryland (Metropolitan Washington DC), is seeking a POSTDOCTORAL RESEARCH ASSOCIATE, (Research Bioinformaticist/Computational Biologist) for a TWO YEAR APPOINTMENT. Salary is commensurate with experience ($55,706 - $72,421 per annum) plus benefits. Citizenship restrictions apply.

As a member of a multidisciplinary team of physiologist, molecular biologists, and computational biologists, the incumbent will be working on a rumen metagenomics project. The incumbent's specific responsibilities will include, but not limited to, the following: (1) Implementing algorithms for clustering, alignment, annotation, and gene prediction from metagenomic sequence data. (2) Optimizing sequence assembly algorithms and protocols for short reads, and (3) Developing models to characterize rumen microorganism dynamics in response to nutritional changes.

Full announcement:
http://bifx.org/forums/forum.php?forum_id=6338



Upcoming events



Perl for Biologists, Level 1; Monday, 7 April » Friday, 11 April

- Taught in the context of biological research, this course helps biologists learn how to use the scripting language Perl to automate certain tasks.
http://wiki.bioinformatics.org/CS101A_Perl_for_Biologists,_Level_1

R for Biologists, Level 1; Monday, 14 April » Friday, 18 April

- Taught in the context of biological research, this course helps biologists learn how to use the statistical scripting language R for data analysis. Unlike Bioinformatics Organization course MA101A (Biostatistics with R, Part 1), this course focuses on the syntax and functionality of the language R and may be used in preparation for MA101A.
http://wiki.bioinformatics.org/CS101B_R_for_Biologists,_Level_1

CS102A Perl for Biologists, Level 2; Monday, 21 April » Friday, 25 April

-Taught in the context of biological research, this course shows biologists how to use the scripting language Perl to automate certain tasks. It is a continuation of CS101A Perl for Biologists, Level 1 and covers advanced topics and projects.
http://wiki.bioinformatics.org/CS102A_Perl_for_Biologists,_Level_2

Bio-IT World Conference and Expo; Boston, MA; Monday, 28 April » Wednesday, 30 April

- Location of the Bioinformatics.Org Annual Meeting and presentation of the Benjamin Franklin Award. This premier event demonstrates how technology is transforming the life sciences throughout all discovery and development processes, including over 100 speakers. In addition to affording you access to these high quality talks this event will include an exhibit hall comprised of over 50 booths providing another great way to maximize your participation.
http://www.bio-itworldexpo.com/

CS102B R for Biologists, Level 2; Monday, 5 May » Friday, 9 May

-Taught in the context of biological research, this course helps biologists understand the data analysis and visualization language R. It is a continuation of CS101B R for Biologists, Level 1 and covers advanced topics and projects.
http://wiki.bioinformatics.org/CS102B_R_for_Biologists,_Level_2

BI201A Gene Expression Analysis; Monday, 12 May » Friday, 16 May

-This course helps to demystify Affymetrix analysis so that any researcher can take the basic steps to go from a chip image to a list of genes that are up- or down-regulated in an experiment. Various tools will be covered, e.g. GCOS, Excel, MATLAB, and free tools like R and Dchip. It is geared towards researchers who conduct microarray experiments to study genome-wide expression changes and understand the underlying mechanisms of gene regulation in samples of interest.
http://wiki.bioinformatics.org/BI201A_Gene_Expression_Analysis

Bioinformatics™
Organization

17 Pope Street
Hudson, MA 01749
USA