Textual Description of webbioc

Colin A. Smith

April 3, 2013

Introduction

`webbioc` is a web interface for some of the Bioconductor microarray analysis packages. It is designed to be installed at local sites as a shared bioinformatics resource. Unfortunately, `webbioc` currently provides only Affymetrix-based analysis. (We would certainly like to collaborate with someone interested in creating a cDNA module.) The existing modules provide a workflow that takes users from CEL files to differential expression with multiple hypothesis testing control, and finally to metadata annotation of the gene lists.

1 User Interface Goals

This package aims to address a number of issues facing prospective users:

- *Ease of use.* Using the web interface, the user does not need to know how to use either a command line interface or the R language. Depending on the computer savvy of the user, R tends to have a somewhat steep learning curve. `webbioc` has a very short learning curve and should be usable by any biologist.
- *Ease of installation.* After an initial installation by a system administrator, there is no need to install additional software on user computers. Installing and maintaining an R installation with all the Bioconductor packages can be a daunting task, often best suited to a system administrator. Using `webbioc`, only one such installation needs to be maintained.
- *Discoverability.* Graphical user interfaces are significantly more discoverable than command line interfaces. That is, a user browsing around a software package is much more likely to discover and use new features if they are graphically presented. Additionally, a unified user interface for the different Bioconductor packages can help show how they can be used together in a data-processing pipeline. Ideally, a user should be able to start using the web interface without reading external documentation.

- *Documentation.* Embedding context-sensitive online help into the interface helps first-time users make good decisions about which statistical approaches to take. Because of its power, Bioconductor includes a myriad of options for analysis. Helping the novice statistician wade through that pool of choices is an important aspect of `webbioc`.

2 Architecture

`webbioc` is written in a combination of Perl, R, and shell scripts. For most processing, the Perl-driven web interface dynamically creates an R script, which is processed in batch mode. A shell script controls the execution of R and catches any errors that result.

The web interface can be configured to execute the shell script in one of two ways. In the single-machine configuration, Perl forks off an auxiliary thread which then runs the script. However, `webbioc` can also be configured to use PBS (Portable Batch System) to send the job to another computer for processing.

If used in a cluster configuration, `webbioc` depends on having a shared partition between the web server and all the compute nodes. This is typically accomplished with NFS. The shared partition allows the compute nodes to push results directly out to the user without directly interacting with the web server.

Microarray analysis is very data intensive and tends to produce large files. Thus special care must be taken to efficiently move that data back and forth between the web-client and the server. `webbioc` uses two different systems for exchanging data with the client, one for input and the other for output.

The Upload Manager handles all files to be processed by Bioconductor. When users start a session with the upload manager, they are granted a unique token that identifies the session. Using that short random string of letters and numbers, they may access their uploaded files from any of the Bioconductor tools. In that way, users must only upload files a single time. Upload manager sessions are meant to be temporary, with any files being automatically deleted after a given amount of time.

For output of results, `webbioc` creates static HTML pages on the fly that contain relevant images or files. Like the upload manager, job results are uniquely identified and are only temporarily stored. Because saving old results could potentially fill up the disk, those too should be automatically purged on a regular basis.

The results of one job may need to be fed into the input of another. For instance, expression summary data created by `affy` needs to be supplied to `multtest` for detection of differentially expressed genes. To facilitate that data exchange, the web interface will optionally copy results back to the upload manager for processing by other packages.

The data interchange format used by `webbioc` is the standard R data file. By convention, each file contains only one object. Additionally, instead of using the common .Rda or .Rdata extensions, `webbioc` will use the class of the stored object as the file extension. This provides a useful abstraction that many computer users have come to understand and expect. The extensions are merely for the benefit of the user as the web interface ignores them.

3 System Requirements

- **R 1.8.0:** `webbioc` has only been tested with R 1.8.0 and later and there is no guarantee that it will work with any version earlier than that.
- **Biobase, affy, multtest, annaffy, vsn, gcrma, qvalue:** `webbioc` directly invokes these packages for processing work. Additionally, those packages depend on others that are not listed here but must also be installed.
- **Bioconductor metadata packages:** Much of the computation done by `affy`, `annaffy`, and `gcrma` depends on pre-built metadata packages available on the Bioconductor web site. We recommend that all of the metadata packages be installed so that users are given the maximum flexibility with which chips they may process. As of this writing, a full install of Bioconductor and all data packages uses approximately 1.25 GB of disk space.
- **Unix:** `webbioc` was written in a Unix environment and depends on many Unix conventions including path names, directory structure, and interprocess communication. It is doubtful whether it will run under Windows and has not been tested as such. However, an ambitious person could port `webbioc` to Windows. The community would greatly appreciate such an effort.
- **Perl 5.6:** Development and testing has been done with both Perl 5.6 and 5.8. It will likely work with either. Perl 5.6 may however require installation of some modules that do not ship as part of the default installation. `webbioc` currently uses the following modules: CGI, Digest::MD5, File::stat, IPC::Open3, and POSIX.
- **Ghostscript:** The web interface uses Ghostscript to produce raw graphics. It must be installed.
- **Netpbm:** The Netpbm series of programs handle graphics manipulation. If you install it using the RPMs, make sure to install both netpbm and netpbmprogs.
- **SGE or PBS (Optional):** `webbioc` has been developed and tested with two batch queueing systems, the Sun Grid Engine and the Portable Batch System. (Only PBS Pro has been tested, although OpenPBS should also work.) If you will be using a batch queueing system, `webbioc` depends on having a shared filesystem mounted in the same place on the web server as well as all the compute nodes. Please note that these are optional as `webbioc` can also run jobs by forking to the background. Adding support for another batch queueing system is a fairly trivial matter. Please contact the author if you are interested in using a different system.

4 Installation

Installing the `webbioc` files is relatively straightforward. First create a directory within the web server's CGI directory that will hold all the Perl scripts. One might typically call that directory "bioconductor". Copy all the files from /`R lib dir`/webbioc/cgi/ into that directory.

Next verify that the permissions are correct on the copied files. Change go to the directory to which you copied the files. Use the command `chmod 755 *.cgi` to make the CGI scripts executable and the command `chmod 644 *.pm` to make the support modules readable.

You must decide how you want to link the Bioconductor web interface into your existing web site. It was designed to integrate seamlessly into an existing site design. `webbioc` includes a very rudimentary home page in /`R lib dir`/webbioc/www/. You may either use this as a starting point or create your own. Please note that you may have to change the links slightly depending on where you place the CGI scripts.

The last step of the installation is to create two directories where the web interface can store files. Remember that if using a batch queueing system, these two directories must be shared between the web server and all compute nodes via NFS (or some other file sharing mechanism). If other users have access to any of the machines running the web interface, we recommend setting the permissions of these directories so that only the web sever user can read them.

The first directory will be for storing uploaded files. The largest type of file uploaded will probably be CEL files. They are typically around 10 MB each. Therefore, depending on expected server usage, the directory should be kept on a partition with hundreds of megabytes to gigabytes of free space. It can be stored anywhere and does not necessarily have to be web-accessible.

The second directory will be used for storing the results of jobs that clients submit. Job results will typically be anywhere from 5 MB to 5 KB. The free space necessary for this directory is again subject to usage. This directory must be accessible via the web server to allow results to be delivered asynchronously.

Both directories should be regularly purged of old files. In the future we hope to provide scripts that will help you do that. Until then, you will have to handle that yourself. We would appreciate the contribution of any scripts towards that end.

To facilitate installation and updating of metadata packages, `webbioc` includes a function to download and install every metadata package from the Bioconductor web site. The following will install all metadata packages and update any out-of-date metadata packages. (Change the path name to match your system.)

```
library(webbioc)
installReps("/library/install/path")
```

Make sure to run R with a user who has permission to write to your library directory. Depending on your site, you may wish to set up a cron job to execute this code approximately once per month to check for updates or additions to the medadata packages. If packages are already up-to-date, it will not waste bandwidth nor CPU by re-installing them.

5 Configuration

Beyond putting the CGI scripts and HTML page in the right places and setting up directories to receive files, all configuration is done through the Site.pm file. The configuration options are discussed here.

- **UPLOAD_DIR** This is the directory where uploaded files are stored. You should set this to the absolute path name of the upload directory created earlier.
- **RESULT_DIR** This is the directory where the web interface will place all result files. You should set this to the absolute path name of the results directory created earlier.
- **RESULT_URL** This is the absolute URL from which the above directory can be accessed through your web server. In almost all cases this is different than the filesystem path.
- **BIOC_URL** This is the absolute URL to the CGI scripts installed earlier.
- **SITE_URL** This is the site URL including only the domain name. It combined with the above URL to create links within the system.
- **R_BINARY** The path to the R executable you wish to use with the web interface. If using PBS, R must be placed in the same location on both the web server and computed nodes.
- **R_LIBS** If you store R libraries outside the default location, enter the paths to those directories here as a colon-delimited list. Otherwise leave this as an empty string.
- **DEBUG** Turn on or off debug mode. If debug mode is on, the web interface will leave scripts, output, and other files behind for inspection. Otherwise, those files will be deleted before a job completes.
- **SH_HEADER** This option is for appending lines to the top of all shell scripts. It can be very useful for setting environment variables such as PATH. (PATH must include the directories containing standard shell tools, Ghostscript, Netpbm binaries, and sendmail.) Depending on your installation you may have to set the GS_LIB variable so Ghostscript can find its libraries and fonts.
- **BATCH_SYSTEM** This option selects the type of batch queueing system to use. Currently supported options are fork, sge, and pbs.

- **BATCH_ENV** This hash allows you to set environment variables necessary for running the batch queueing system. SGE typically needs SGE_ROOT and possibly SGE_CELL or COMMD_PORT. PBS typically needs PBS_HOME, PBS_EXEC, and PBS_SERVER.

- **BATCH_BIN** This specifies the location of the batch queueing system bin directory.

- **BATCH_ARG** This specifies any additional arguments to be passed to the job submission program that are necessary for job routing, accounting, etc. It can be left blank.

- **site_header** This subroutine is called to produce the site header HTML for all CGI pages. It is implemented as a function to allow you to include other files and/or do any other crazy programming you wish. If you make changes, make sure to set the title of the page to the $title Perl variable.

- **site_footer** This subroutine is called to produce the site footer HTML for all CGI pages. It is your responsibility to close off any HTML tags here that you open in site_footer.