Genetics, Development and Cell Biology
Publications

Genetics, Development and Cell Biology

4-2008

explorase: Multivariate Exploratory Analysis and Visualization for Systems Biology

Michael Lawrence
Fred Hutchinson Cancer Research Center

Dianne Cook
Iowa State University, dicook@iastate.edu

Eun-Kyung Lee
Ulsan University

Heather Babka
Iowa State University

Eve S. Wurtele
Iowa State University, mash@iastate.edu

explorase: Multivariate Exploratory Analysis and Visualization for Systems Biology

Michael Lawrence
Fred Hutchinson Cancer Research Center

Dianne Cook
Iowa State University

Eun-Kyung Lee
Ulsan University

Heather Babka
Iowa State University

Eve Syrkin Wurtele
Iowa State University

Abstract

The datasets being produced by high-throughput biological experiments, such as microarrays, have forced biologists to turn to sophisticated statistical analysis and visualization tools in order to understand their data. We address the particular need for an open-source exploratory data analysis tool that applies numerical methods in coordination with interactive graphics to the analysis of experimental data. The software package, known as **explorase**, provides a graphical user interface (GUI) on top of the R platform for statistical computing and the **GGobi** software for multivariate interactive graphics. The GUI is designed for use by biologists, many of whom are unfamiliar with the R language. It displays metadata about experimental design and biological entities in tables that are sortable and filterable. There are menu shortcuts to the analysis methods implemented in R, including graphical interfaces to linear modeling tools. The GUI is linked to data plots in **GGobi** through a brush tool that simultaneously colors rows in the entity information table and points in the **GGobi** plots.

explorase is an R package publicly available from **Bioconductor** and is a tool in the **MetNet** platform for the analysis of systems biology data.

Keywords: bioconductor, bioinformatics, microarray, graphical user interface, exploratory data analysis, interactive graphics, visualization, metabolomics, proteomics.

1. Introduction

In recognition of the need for biologists to analyze multidimensional, high-throughput data, we have developed a software tool with the following goals:

- Support exploratory analysis of experimental data through the integration of numerical methods and interactive graphics.

- Leverage biological information in the analysis of experimental data.

- Provide a graphical user interface (GUI) on top of statistical methods so that they are accessible to biologists.

High-throughput experiments have become commonplace in biology. The popular microarray measures the levels of tens of thousands of gene transcripts at once. GC-MS and other analytical methods currently have the potential to detect hundreds of metabolite levels, providing a snapshot of the metabolism in a cell. The **explorase** package has been developed for analyzing datasets with measurements on tens of thousands of biological entities, such as genes or metabolites. The **explorase** interface and analysis algorithms are designed for experiments with up to approximately 50 samples (columns in the experimental data matrix).

The untargeted nature of high-throughput experiments pairs well with an approach to data analysis that remains open to the unexpected and allows the analyst to form hypotheses during analysis. The intent is to facilitate the search for interesting features in the data, such as differentially expressed genes or metabolites that follow a similar pattern. This approach is known generally as exploratory data analysis and is the main philosophy behind the design of our software tool.

Interactive graphics are generally useful in exploratory analysis, and they are particularly applicable to analyzing experimental data for several reasons:

- Experimental datasets are multivariate and so benefit from the ability to view different combinations of variables in different ways, simultaneously.

- Results from numerical methods can be interpreted and validated by relating them back to the original data through visual cues.

- Linked interaction can relate individual measurements to the biological system by integrating plots of experimental data with drawings of biochemical networks.

There is a number of software projects for analyzing data from high-throughput biological experiments. One of these is **Bioconductor** (Gentleman *et al.* 2004), a free collection of R packages (R Development Core Team 2008) that provide numerical and graphical methods for helping biologists comprehend their data. There are packages in **Bioconductor** for analyzing and visualizing networks and various types of experimental data, including microarray and mass spectrometry data.

While R and the **Bioconductor** project provide sufficient support for numerical and general graphical methods, the R graphics system is not designed for interactivity. A specialized application for multivariate interactive graphics, **GGobi** (Swayne *et al.* 2003), has been interfaced with R through the package **rggobi** (Temple Lang and Swayne 2001; Temple Lang *et al.* 2008). **GGobi** provides interactive scatterplots, parallel coordinate plots, barcharts and other types of displays. Edges may be displayed in scatterplots in order to draw networks. Interaction modes include linked brushing between plots by identifier and categorical variable, point and edge querying, and pan/zoom.

The complementary nature of R/**Bioconductor** and **GGobi** encourages their combined application to the analysis of biological data. However, this is hindered by their lack of lack of accessibility to biologists. All R packages, including those in **Bioconductor**, are driven through the R language, which facilitates their application to a wide variety of data analysis tasks. The flexibility and expressiveness of a script-driven interface is a double-edged sword, however. Biologists unfamiliar with programming and command-line interfaces struggle to take advantage of R and **Bioconductor** packages that lack a GUI. **GGobi** is designed to be flexible and open-ended, with the goal of supporting a wide range of analyses. However, this generality means that the biologist receives no biology-specific guidance during data analysis and visualization tasks.

We have developed a GUI-driven R package named **explorase** for the graphics-intensive exploratory analysis of biological experimental data. It is a tool in the **MetNet** platform (http://www.metnetdb.org/, Wurtele *et al.* 2003, 2007) for systems biology data analysis and is publicly available from **Bioconductor** (http://www.bioconductor.org/) as of **Bioconductor** version 2.1.

The **explorase** package has the following general features:

- A collection of numerical methods, such as distance measures and linear models, implemented in the R language (R Development Core Team 2008).

- Linked data plots based on the **GGobi** tool for multivariate interactive graphics (Swayne *et al.* 2003).

- A GUI, designed in collaboration with biologists, that integrates numerical methods with graphics and provides general features such as the loading and subsetting of data.

Section 2 presents an overview of the software and the next section describes the numerical and graphical methods it provides. The layout of its GUI is detailed in Section 4. This is followed by a tutorial that guides the user through a hypothetical analysis. Finally, the paper will conclude by discussing the future of **explorase**.

2. Overview

As its name suggests, **explorase** is designed to facilitate the exploratory analysis of biological data by combining the numerical methods of R (and **Bioconductor**) with the graphical methods of **GGobi**. Most users will only interact with **explorase** through its GUI, and the GUI and plots of **GGobi**. The **explorase** GUI is shown in Figure 1.

The user begins an **explorase** session by loading information describing one or more experiments. This includes the actual experimental measurements, as well as metadata and other supporting information. Each type of information is described below:

Experimental data matrix Measurements of the levels of transcripts, metabolites or some other biological entity for every sample in the experiment.

Experimental design matrix Description of experimental design, factors such as genotype or time. Each sample is labeled by the factor levels of the experimental design.
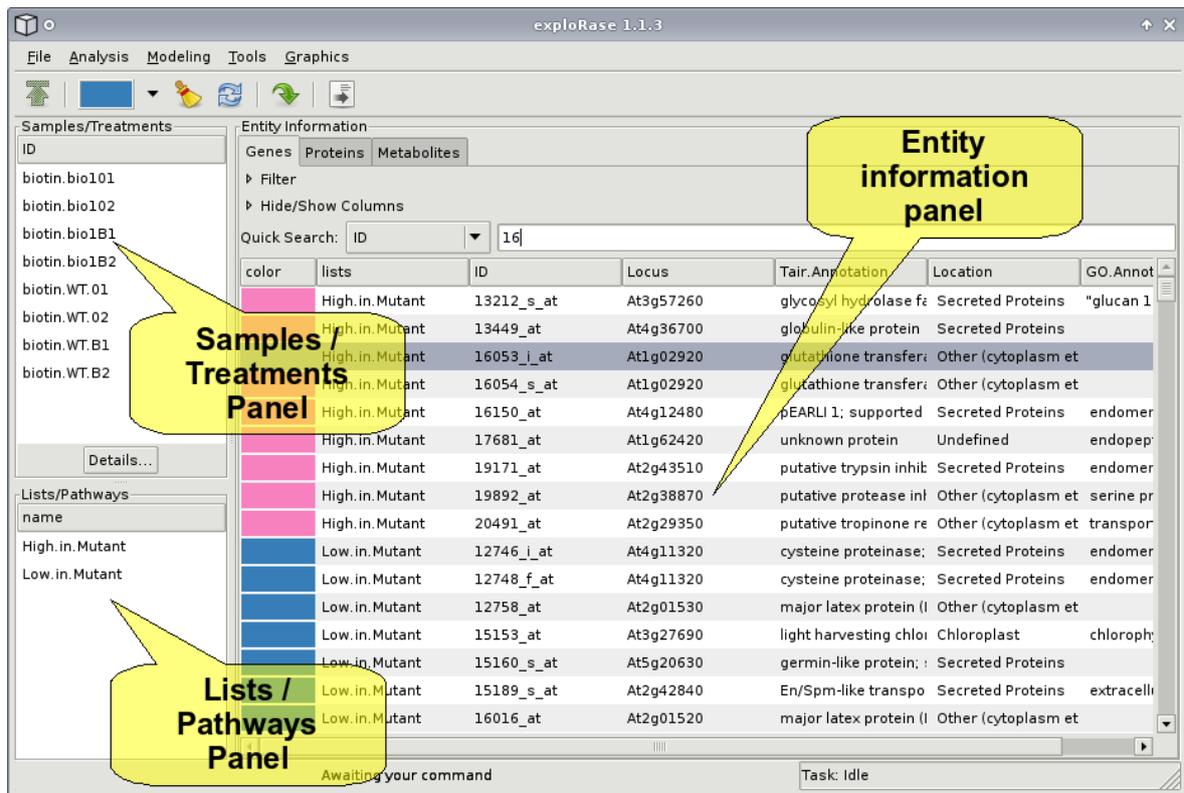
Figure 1: The main GUI of **explorase** with the three main panels denoted by the yellow blurbs. The `Entity information` panel, located on the right in the GUI, is dominated by the entity information table, which contains metadata and analysis results for each entity in the experiment. Above the table is a quick search bar and tools for filtering the rows and columns of the table. To the left of the `Entity information` panel are two panels. The top one is the `Samples/Treatments` panel and lists the samples, such as microarray chips, in an experiment. Below is the `Lists/Pathways` panel, which contains the names of user-created entity lists. At the top of the GUI are the toolbar and menubar with options for loading and saving the data, analyzing the data and performing other functions. Please note that the appearance of the GUI (especially the button icons) depends on the **GTK+** theme, so there may be some aesthetic differences between the screenshots in this paper and a user installation of **explorase**.

Entity annotation matrix Annotations, such as GO terms, of biological entities measured in the experiment.

Entity list User-defined lists of entities of interest, such as lists of metabolites in a common pathway or co-expressed genes.

While no particular type of information is required to run **explorase**, each feature has specific data requirements. For example, the linear modeling features require an experimental design matrix and the experimental data.

The data from a set of related experiments are organized into a *project*. The user creates a

project by importing individual data files into an empty **explorase** session. In future sessions, the user only needs to load the project to begin the analysis. Only one project may be loaded per session, and a project may contain at most one experiment for each type of biological entity. The entity types built into **explorase** are genes, metabolites and proteins; expert users may define custom types through the R command line interface. The distinction between entity types is currently only for organizational purposes, though it may gain meaning in the future.

The files in a project are physically organized into a directory (folder) in the file system. The user need only specify the directory to load every file (i.e., experimental data files and metadata files), in the project at once. The format of every file is CSV, which is compatible with most spreadsheet applications. The type of information contained in a file and the type of biological entity the information describes are indicated by the filename extension.

Once a dataset and related information is loaded into **explorase**, the user may proceed with the analysis by performing operations such as:

- Subset the observations by criteria such as the minimum fold change and maximum variance across replicates.

- Check the quality of the data with **GGobi** graphics, such as scatterplots, scatterplot matrices and parallel coordinate plots.

- Browse entity metadata and analysis results in a filterable, searchable and sortable table.

- Color rows in the entity information table and the corresponding points in **GGobi** plots using the brush tool.

- Detect differentially expressed genes through the graphical interface to the **limma** package (Smyth 2005) and view their profiles in a **GGobi** parallel coordinate plot.

- Find patterns in the data through hierarchical clustering and the explicit pattern query tool.

- Export analysis results and lists of interesting entities as CSV files.

3. Methods

3.1. Numerical methods

The numerical methods in **explorase** are designed to assist in finding biological entities with patterns that depend on experimental conditions or are similar to the pattern of a given entity or user-specified pattern. It is also possible to cluster entities according to a selected distance measure.

Finding entities with interesting patterns

One means of finding interesting patterns with **explorase** is to compare two replicates or replicate means by a selected distance measure. The supported distance measures are difference, residuals from regressing one condition against the other, angle between the diagonal

and the line from the origin to the point in the scatterplot of the two variables, and the Mahalanobis distance (e.g., Johnson and Wichern 2002). Of these, the difference is the simplest and the most often applied to transcriptomics data. A distance measure may be use to check the agreement between two replicates or to evaluate differences in entity levels between treatments using replicate means.

Linear modeling is a more sophisticated technique for estimating the effects of experimental conditions on each entity. The **explorase** GUI includes an interface to **limma** (Smyth 2005) that fits a linear model to each entity and shrinks the variance using empirical Bayes analysis to yield p value estimates that tend to correspond to what a biologist considers interesting: entities with relatively high basal levels as well as significant difference across conditions. The significance of each user-selected experimental design factor, as well as their interactions, is estimated without the user needing to manually specify any contrasts. The output of the **limma** tool may consist of the raw p values, p values corrected by FDR or another method, the corresponding F statistics, the contrast coefficients and the fitted values. The **limma** interface has an advanced feature for fitting contrasts that evaluate whether an entity pattern is linearly or quadratically dependent on time in time-course experiments.

There is also a separate polynomial model for estimating the effect of time. Its critical difference from **limma** is that it accounts for the order of the time points and thus is likely more realistic for time-course experiments. The user also can apply this polynomial model to evaluate interaction effects between time and other factors. The output contains the p value and coefficient for each effect, as well as the F statistic and the sum of squares error for the overall model.

Searching for entities with specific patterns

The **explorase** package offers several distance measures for comparing entity patterns. These include cosine angle (uncentered correlation), Euclidean, Pearson correlation and Canberra (e.g., Johnson and Wichern 2002). The Euclidean distance is usually not of interest, as it is based on the magnitude of the levels, not their pattern; however, it may be useful for finding entities that are present at similar levels over the course of the experiment. The Pearson correlation disregards magnitude, so it may be useful for identifying entities with similar patterns regardless of their levels. This could, for example, help identify gene regulatory interactions, where the increase (or decrease) in one transcript results in the decrease (or increase) in another. In contrast, the cosine angle distance considers both pattern and magnitude and thus may be useful if the biologist wishes to focus on entities present at similar levels while still considering the pattern. If the level of the query entity is relatively high (i.e., above background noise), the cosine angle measure reduces the number of hits against entities that are present only at low (i.e., background) levels and may thus be considered uninteresting. The Canberra distance may be particularly useful for metabolomics data as it is numerically stable when faced with zero values, which result from the common practice of imputing non-detects as zeros.

The user may also search for an explicit pattern by specifying "Up", "Down" or "Same" for each transition between adjacent samples in a user-specified list. The entity patterns are matched to the query by denoting a transition as "Up" if the transition value (difference between the pair of samples) is above the q quantile and "Down" if the value is below the $1 - q$ quantile. Everything between the quantiles is marked as "Same". The parameter $q \in [0, 1]$ is specified

by the user with a slider.

Clustering of entities

For clustering the observations, **explorase** performs agglomerative hierarchical clustering using Ward's linkage method (e.g., Johnson and Wichern 2002). The user can choose from the several distance measures: cosine angle, Euclidean, Pearson correlation and Canberra. These are the same distance measures described for comparing entity patterns against a query entity pattern. A very similar question is asked when clustering, so the same criteria apply for selecting a distance measure. One of the major differences between comparing entitity patterns and clustering is that in clustering the distance is calculated between every pair of entities rather than between one entity and the others. Given the size of high-throughput datasets, it is compuationally intensive to calculate the pairwise distance for every entity in the experiment, and the results can be difficult to interpret. Thus, it is recommended that the user select a small subset of entities for clustering. **explorase** supports efficient clustering of up to approximately 50 entities.

3.2. Graphical methods

The graphics in **explorase** are interactive and designed for exploratory data analysis. The data plots are displayed by **GGobi**. For visualizing the experimental data, **explorase** provides scatterplots, scatterplot matrices, parallel coordinate plots and histograms.

Scatterplots

The scatterplot, or the basic X vs. Y plot, is the most generally applicable **explorase** plot type. For example, the user may compare two samples, such as a pair of replicates, or two conditions, averaged over the replicates. In this way, the scatterplot is the graphical equivalent of distance measures for comparing the levels of an entity between two samples or conditions. The scatterplot is also useful for interpreting analysis results. For example, comparing the p value for a **limma** effect with its coefficient, as in Figure 2, is particularly effective for detecting patterns that are both significant and of high amplitude.

Scatterplot matrices

The scatterplot matrix is a symmetric grid of scatterplots. Every row has the same variable on the vertical axis and every column shares the variable on the horizontal axis. Along the diagonal, the X and Y variables are the same, so histogram is displayed for that variable. Scatterplot matrices are useful for obtaining an overview of the data, because multiple variables are compared at once. One common application is the comparison of replicate sets during data quality checking.

Parallel coordinate plots

The parallel coordinate plot displays a profile of each entity across a sequence of variables, normally the samples or replicate averages. Due to the large number of entities in microarray, proteomics and metabolomics experiments, the profiles tend to be overplotted to the extent that individual profiles are not recognizable (e.g., the grey profiles in the parallel coordinate plot in Figure 2). Linked brushing helps to overcome to this problem. For example, as shown
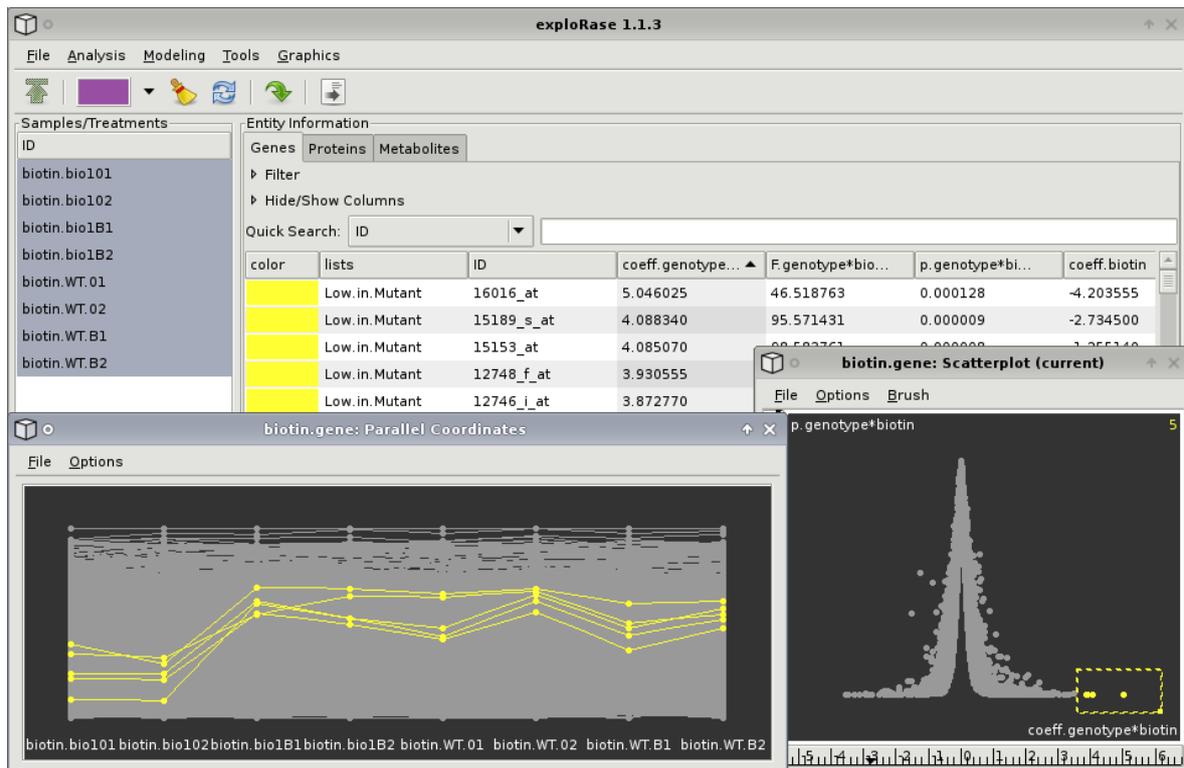
Figure 2: Visualizing and validating Limma results with **explorase**. The dataset is from the microarray experiment described in Section 6 (Cook *et al.* 2007). On the right is a scatterplot of p value versus coefficient for the genotype contrast fit by Limma. By highlighting a point in the scatterplot, the corresponding profile in the parallel coordinate plot on the left is also highlighted.

in Figure 2, the user could brush outlying points in the scatterplot of p value versus coefficient to highlight the corresponding profiles in the parallel coordinate plot. This permits the visual validation of patterns deemed significant by numerical methods.

Histograms

The interactive histogram is another tool for interpreting analysis results and checking data quality. The user can view profiles that are outliers with respect to a statistic by brushing outlying points in the histogram of the statistic. For example, assume the user has brushed a particular gene of interest and wishes to view profiles that are similar to that of the chosen gene. The user could calculate a distance measure between the gene and the others and compare profiles in a parallel coordinate plot by brushing the outlying points (with a new color) in a histogram of the distances.

4. GUI features

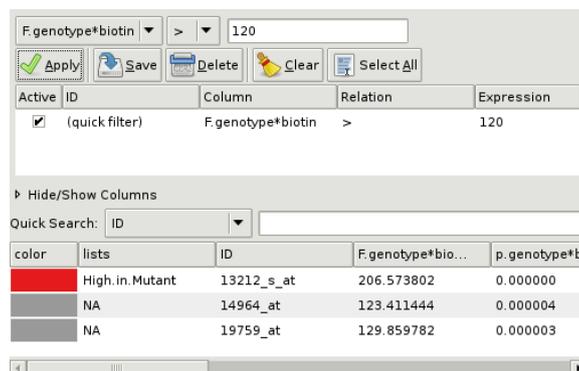The main GUI of **explorase**, shown in Figure 1, consists of three panels (`Entity information`,

Figure 3: The **explorase** filter GUI that expands above the entity information table. The active filter rule accepts only the genes with a *F.genotype* ∗ *biotin* value greater than 120.

`Samples/Treatments` and `Lists/Pathways`), a toolbar and a menubar. The primary design considerations for the GUI are simplicity and usability. There is no attempt to completely map the features of the underlying tools to the **explorase** GUI. Rather, the GUI supports a subset of the features most useful in the analysis of high-throughput transcriptomics, proteomics and metabolomics data and augments this subset with shortcuts and conveniences for biological data analysis. The GUI has been designed in collaboration with biologists, to help ensure that **explorase** is accessible to those who are performing the experiments and generating the data.

4.1. Main panels

`Entity information` *panel*

The largest panel contains the entity information notebook, which has a tab for each entity type of interest. The default entity types are genes, proteins, and metabolites; new types are easily added by expert users through the **explorase** R API. Each tab contains a table containing entity information (metadata and analysis results), an expandable panel for filtering the table rows, an expandable panel for hiding or showing table columns and an entry box for searching the table.

The table has a row for each biological entity in the experimental data. The columns of the table contain metadata, such as biological function and biochemical pathway membership. There are two special built-in columns at the left. The first displays the color of the entity chosen by the user. This color matches the color of the glyphs for the entity in the **GGobi** plots. The other column indicates the user-defined entity lists to which the entity belongs. The table may be sorted according to a particular column by clicking on the header for the column.

A filtering component, shown in Figure 3, is made visible by clicking on the `Filter` label above the table in the `Entity information` panel. This filters the entity information table, as well as the **GGobi** plots, by any column in the table. Columns containing text data may be filtered according to whether a cell value equals, starts with, ends with, contains, or lacks a user-input text phrase. Regular expression matching (Friedl 2006) is also supported. Numeric
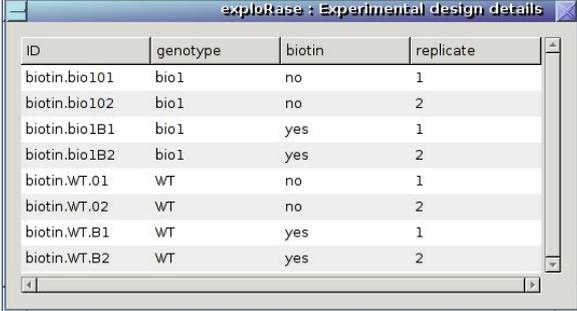
Figure 4: The experimental design table, with a column for each factor and a row for each condition.

values may be tested for being greater than, less than, equal to, or not equal to a user-input number. When filtering by color, the user may choose from the current palette of colors. It is also possible to filter by entity list membership, so that only the entities that belong to a specified list are included in the table. After applying a rule, it may be saved. The saved rules are displayed in a table below the rule editor. There is a checkbox in each row that toggles the activation state of the rule. Buttons allow the deletion of selected rules and the batch activation and deactivation of every rule. Only those cases that pass the intersection of all active filter rules are displayed in the entity information table and the **GGobi** plots. After filtering, the user might select all the visible entities and save them to an entity list for future recall.

Clicking on the `Hide/show columns` label in the `Entity information` panel expands a component that lists the column names of the entity table and allows the user to specify whether each column is hidden or shown. This helps keep the table clean when generating many columns holding analysis results.

Just above the entity information table is a `quick search` bar to search individual columns. The user selects the column to search in the pull-down menu on the left and enters a query in the text box. As the user types, the table scrolls to the first row that matches the query.

`Sample/Treatments` *panel*

To the left of the `Entity information` panel are two panels. The upper one lists the biological samples, such as chips for a microarray experiment, that are in the experimental data. The user may select samples from the list in order to limit the scope of the analysis. It is possible to select a range of samples by holding down the `SHIFT` key and clicking on the end-points of the range. This is particularly useful after the list has been sorted by an experimental factor using the experimental design table described in the next paragraph.

Clicking on the `Details` button below the `Sample/Treatments` list displays a table describing the experimental design, as shown in Figure 4. There is a column for each factor in the experiment, and the rows correspond to conditions. Similar to the `Entity information` panel, clicking on a column header sorts the table, as well as the `Sample/Treatments` list in the main GUI, by that column.

`Lists/Pathways` *panel*

The bottom panel contains user-defined entity lists. These lists store a group of user-selected entities, usually based on the result of an analysis. Selecting an entity list automatically selects the corresponding entities from the list in the entity information tables. The selected entities may then, for example, be brushed or sent as a query to **AtGeneSearch**, as described in Section 4.2.

4.2. Toolbar

Above the main GUI panels is the toolbar (Figure 1), which contains many important buttons. We will describe the buttons from left to right. The button with the folder icon provides a shortcut for loading a project. Perhaps the most important button is the brush tool, which appears as a rectangle filled with the current brush color. Clicking on the brush button colors the selected entities in the currently visible entity information table and the same entities in the **GGobi** plots. The color is selected from a palette that drops down from the button. Besides the brush button are two more brush-related buttons: the first for resetting the colors to the default (gray) and the other for updating the colors in the entity information table to match those in **GGobi**. The next button to the right, shown in the screenshot as a single curved arrow, queries **AtGeneSearch**, a web interface to **MetNetDB** for accessing additional metadata about the selected entities (Wurtele *et al.* 2003, 2007). **AtGeneSearch** provides links to other web data sources. The right-most button creates an entity list from the entities selected in the entity information tables and adds the nascent list to the `Lists/Pathways` panel.

4.3. Menubar

`File` *menu*

At the top of the main **explorase** GUI is the menubar (Figure 1). The `File` menu provides options for loading and saving files and projects. The `Open` item is for loading already-created projects. To load individual files and merge them into an opened project the user may choose the `Import File(s)` item and select the files using a dialog. The `Save` item saves the entire project to a user-specified directory. The user may select an item from the `Export` submenu to save an individual project component, such as the entity information or a selected entity list, to a CSV file.

`Analysis` *menu*

The `Analysis` menu lists a collection of numerical analysis methods, as described in 3.1. The first set of methods consists of distance measures for comparing the levels of each entity between two samples or conditions. The next set of methods are distance measures for comparing a selected entity against the rest. The final two methods are hierarchical clustering and pattern finding. The cluster results are displayed in an interactive R plot, shown in Figure 5; clicking on a branch point of this R plot brushes the descendent entities. The results of each analysis are added as a column in the entity information table and as a variable in **GGobi**. This allows the user to sort and filter according to the results of the statistical analyses, as well as visualize these results in **GGobi**.
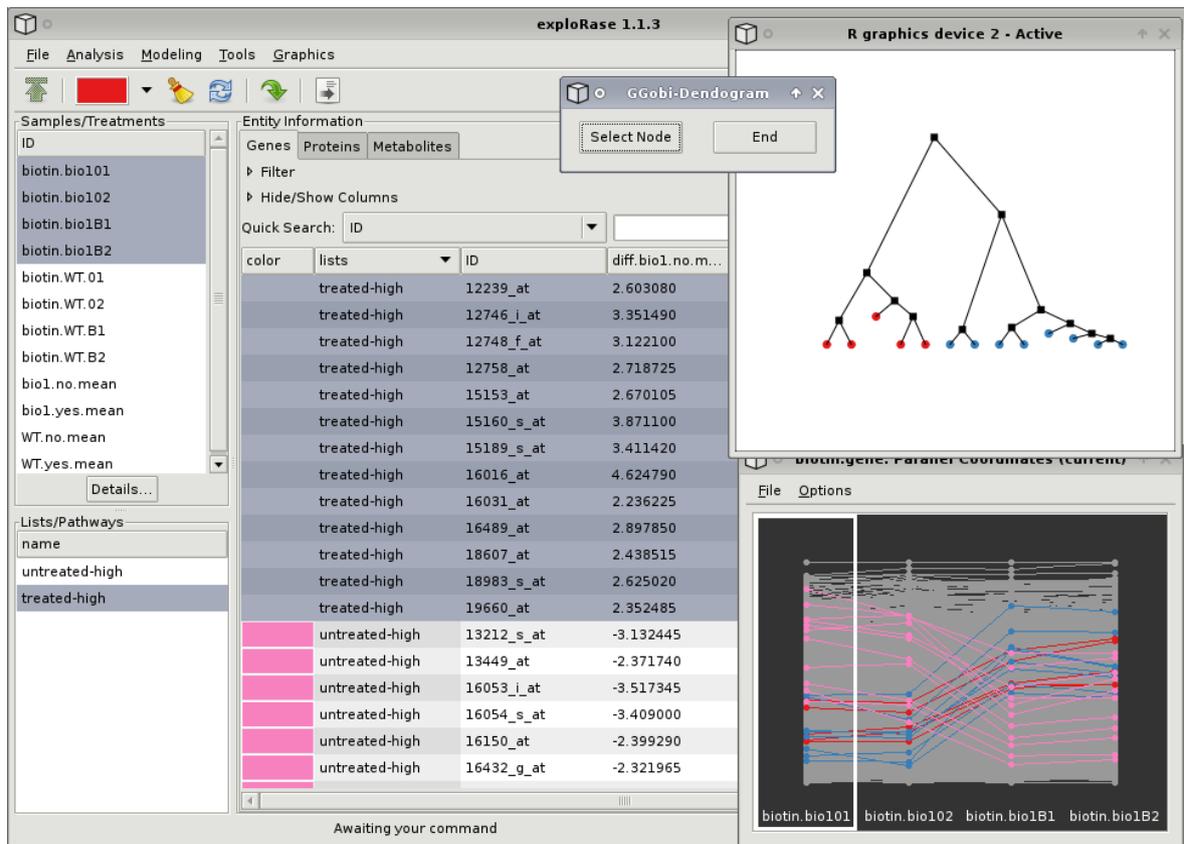
Figure 5: The hierarchical cluster browser. Clicking on a node selects its children in the **explorase** information table. In this screenshot, the "treated-high" genes (originally all colored blue) have been clustered according to the mutant chips. The left child of the root has been clicked, resulting in the coloring of five genes in **explorase** (and **GGobi**) with the current brush color (red). It appears from the parallel coordinate plot that the patterns of the red genes have more in common with each other than with the blue genes.

A unique feature of **explorase** is the pattern finder, which calculates whether an entity is significantly rising or dropping relative to the others for each sample transition. The results are displayed as arrows embedded in the entity information table, as shown in Figure 6. The dialog named `Find Patterns` allows the user to query for specific patterns. To specify a query pattern, the user chooses whether a particular transition should be "Up", "Down" or the "Same" (these match the terms used by the `Find Patterns` dialog). The vertical slider on the left of the dialog specifies the number of entity transitions, centered on the median, that are considered to stay the "Same". All transitions less than the assumed "Same" transitions are considered "Down" and the remaining transitions are considered "Up". When the `Find` button is clicked, the entities with matching patterns are selected in the entity table.

`Modeling` *menu*

The `Modeling` menu launches graphical interfaces to linear modeling tools in R. Both interfaces
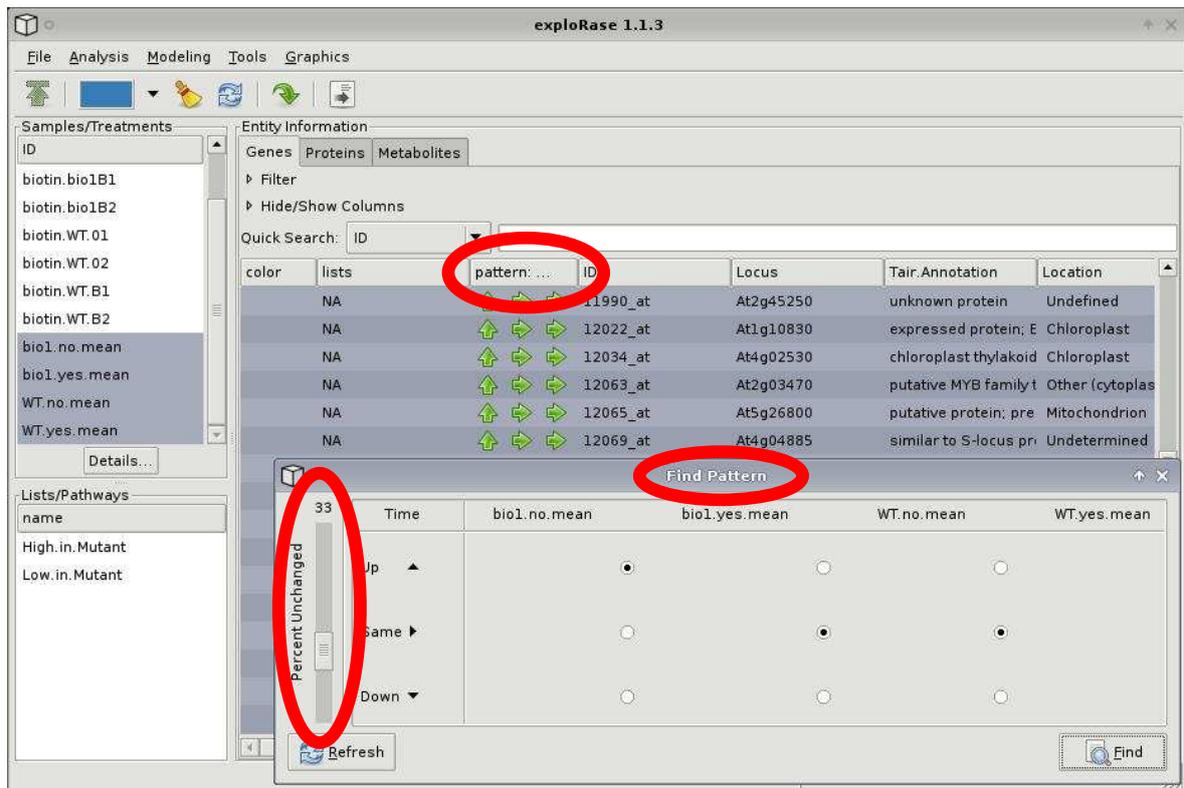
Figure 6: The pattern finder. The calculated patterns across a time course experiment are shown in the "pattern..." column as arrows representing the direction of each transition. The pattern finder dialog selects rows that match the specified pattern. Here, a constantly increasing pattern has been selected. The vertical slider on the left of the dialog specifies the number of entity transitions, centered on the median, that are considered to stay the "Same". All transitions less than the assumed "Same" transitions are considered "Down" and the remaining transitions are considered "Up".

are shown in Figure 7.

The **limma** interface leverages the **limma** package (Smyth 2005) from **Bioconductor** (Gentleman *et al.* 2004). The interface prompts the user for the factors to include in the model, including interactions among factors. The user may also choose which results (p values, corrected p values, F statistics, coefficients, or fitted values) to include in the entity information table and **GGobi**. An `Advanced` drop-down offers additional options, such as the method for p value adjustment and tests for time linearity.

An interface is also provided for time-course modeling. It fits a polynomial model in time. The user may define the degree of the time polynomial and choose whether the time variable should be treated as a quantitative variable (actual) or as an ordinal variable (virtual).

`Tools` *menu*

The `Tools` menu contains methods for processing experimental data. There are items for calculating replicate means, medians or standard deviations and adding them to the data.
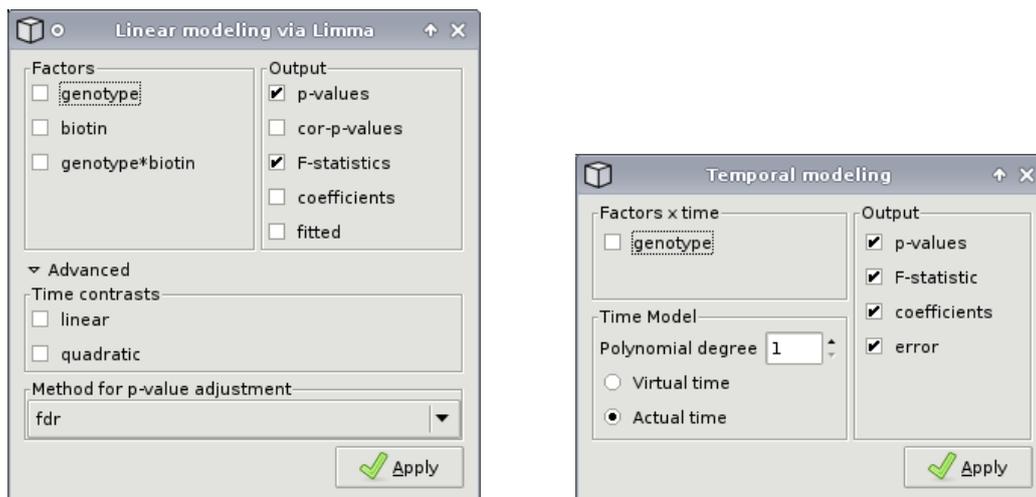
Figure 7: Linear modeling dialogs. On the left is the interface to **limma**, and on the right is the temporal modeling interface. The user may select the factors and outputs of interest, as well as specify various other parameters.
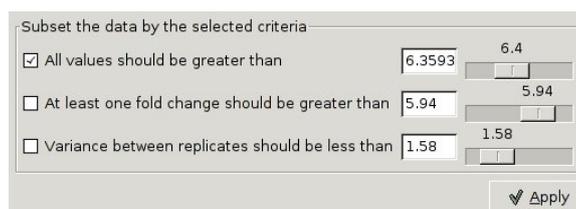

Figure 8: Simple subsetting GUI. The user may activate rules that filter entities based on their level, fold change, and replicate variance.

The means and medians are automatically included in the list of experimental conditions, so that they may be used in numerical analysis. The second option launches the dialog shown in Figure 8 that provides several simple rules for filtering out entities based on the experimental data. The cutoffs are based on minimum value, minimum fold-change, and maximum variance between replicates. This helps the user focus on particular aspects of the data, such as entities that are changing more between treatments than within. The user may enter the test values directly or use the slider to get some idea of the range of values.

5. Getting started

The first step towards analyzing data with **explorase** is to load the data. The **explorase** package has not been designed for data preprocessing, so all preprocessing must be done before loading data into **explorase**. Usually this involves steps like normalizing and log transforming the data. All files read by **explorase** must adhere to the comma separated value (CSV) format, as interpreted by the R CSV parser. This format is compatible with the output of **Bioconductor** tools and the CSV export utility of Microsoft **Excel**. Accordingly, the file

Type	File Extension (+ `.csv`)	Example
Transcriptomic Data	`gene.data`	`mittler.gene.data.csv`
Gene Information	`gene.info`	`affy25k.gene.info.csv`
Gene Exp. Design	`gene.design`	`mittler.gene.design.csv`
Metabolomic Data	`met.data`	`suh-yeon.met.data.csv`
Metabolite Information	`met.info`	`suh-yeon.met.info.csv`
Metabolite Exp. Design	`met.design`	`suh-yeon.met.design.csv`
Proteomic Data	`prot.data`	`some-proteins.prot.data.csv`
Protein Information	`prot.info`	`some-proteins.prot.info.csv`
Protein Exp. Design	`prot.design`	`some-proteins.prot.design.csv`
Interesting Entities	`list`	`favorite-metabolites.list.csv`

Table 1: Mapping from data type to filename extension per the **explorase** filenaming convention. **explorase** requires project files to be named using these file extensions. An example project with correctly formatted files is in the supplemental data and available from the **explorase** website (Lawrence 2007).

containing the matrix of experimental measurements must be formatted as CSV, with the values from each sample (i.e., chips in a microarray experiment) stored as a column. The first row should hold the names of the corresponding samples. The first column, which does not require a name in the first row, should hold unique ids for each biological entity (transcript, protein, etc.) measured in the experiment.

In addition to the experimental measurements, **explorase** supports (and, for some features, requires) several types of metadata, all formatted as CSV. The experimental design matrix is required for linear modeling and aggregating replicates. Like the experimental data, the first row should name the design factors, such as *genotype*, *time*, and *replicate*. Some factor names have special meaning. In particular, *time* is used as a factor in the temporal modeling tool and *replicate* is used in linear modeling and averaging over replicates. Each cell in the first column of the design matrix should match one of the sample names in the experimental data.

Another type of metadata is the entity annotations that are shown in the central table of the **explorase** GUI. The only restriction is that the first column should hold entity identifiers that match those of the experimental data.

Finally, entity lists are stored as one or two column matrices. If two columns are present, the first column is interpreted as the type of the entity, such as *gene*, *prot*, or *met*. This allows storing entities of different types in the same list. The other column holds the identifiers of the entities that belong to the list. The name of that column is the name of the list in the **explorase** GUI.

In order to automatically detect the type of data being loaded, **explorase** expects the input files to be named according to a specific convention. The mapping from data type to filename extension is given in Table 1. The user must ensure that the input files are named according to that convention.

The data loading process is further simplified by support for projects: all of the data files may be placed into an empty folder and loaded in a single step by choosing the folder in the open project dialog. The types of the files are determined by their file extension.

6. Demonstration

In order to briefly demonstrate the features of **explorase**, we consider a microarray dataset from an experiment investigating the response of biotin-deficient Arabidopsis mutants to treatment with exogenous biotin (Cook *et al.* 2007). The mutants were analyzed with and without biotin treatment. Wildtype plants were used as a control and there were two replicates for each set of conditions. Figure 4 summarizes the experimental design. The dataset was normalized using the RMA method.

The first step, after launching **explorase**, is to load the data. One easy way to load data into **explorase** is as a project. Projects are directories in the file system that contain the experimental data, design matrix, entity metadata, entity lists, etc, as files. A zip archive containing an example **explorase** project for the biotin data, with correctly formatted files, is provided on the **explorase** website (Lawrence 2007).

To load the project:

1. Click the `Open` button at the left-end of the toolbar (see Figure 1).

2. In the file open dialog, select the `biotin` directory from the (uncompressed) zip archive and click `Open`.

The primary goal of this example analysis is to determine which genes appear to respond to biotin treatment in the mutant. In order to compare across conditions without having to consider each replicate individually, the replicate mean values should be added to the experimental data, assuming that there are no major inconsistencies within the replicate pairs. To add the means to the data: choose the `Average over the replicates` option from the `Tools` menu.

Figure 9 displays the result of subtracting the untreated mutant mean from the treated mutant mean in the sample dataset. Sorting by the difference column in the information table allows the coloring of the selected extreme rows using the **explorase** brush button. Alternatively, the user could brush the outlying points in the **GGobi** plots and then update the colors in the entity information table to match those in **GGobi**. The genes at each extreme are grouped into entity lists. The genes brushed in pink are those that have higher expression in the untreated plants compared to the treated, while the blue have lower expression in untreated plants.

To color and group the entities with the most extreme differences between treated and untreated mutant means, follow these steps (as illustrated in Figure 9):

1. Select `bio1.no.mean` and `bio1.yes.mean` in `Samples/Treatments` panel (use the `CTRL` key for multiple selections).

2. Open the `Analysis` menu in the menubar at the top of Figure 1. Choose the `Subtract` item from the `Find Difference (two conditions)` submenu.

3. Once the column containing the differences appears in the entity information table (the large table in the center of Figure 1), click on the column header (`diff.bio1...`) until the results are sorted in decreasing order.

4. Select a range of rows at the top of the entity information table (i.e., by holding down the `SHIFT` key).
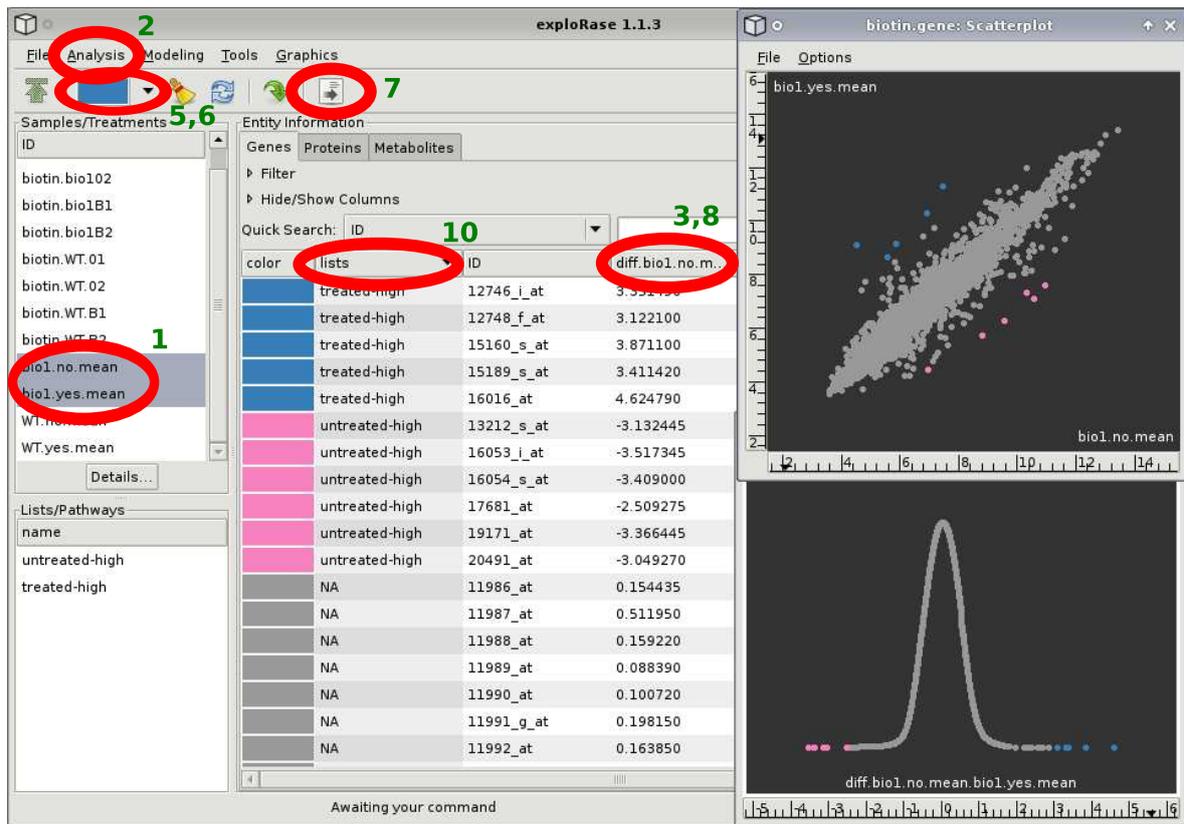
Figure 9: Difference calculation between the biotin mutant with and without external biotin. The **GGobi** scatterplot (above) compares the two conditions; below is a histogram of the difference calculation. The numbered red circles drawn on top of the screenshot illustrate the steps to calculate the differences, as explained in the example.

5. Click on the downward-pointing arrow on the right side of the `Brush` button and select the blue color.

6. Click the `Brush` button to color the selected rows blue.

7. Click the `Create List` button in the toolbar and enter "treated-high" into the entry that appears in the `Lists/Pathways` panel at the bottom-left of Figure 1.

8. Click on the header of the difference column of the entity information table again to resort the rows of the entity table so that the rows are in increasing order.

9. Repeat steps 4-7 but use pink rather than blue as the brush color and enter "untreated-high" when creating the list.

10. To sort the rows in the entity table by their list membership, click on the header of the `List` column in the entity information table.

The **GGobi** scatterplot at the top-right of Figure 9 compares the two means, showing that the colored observations are indeed outliers. Below the scatterplot is a histogram showing the
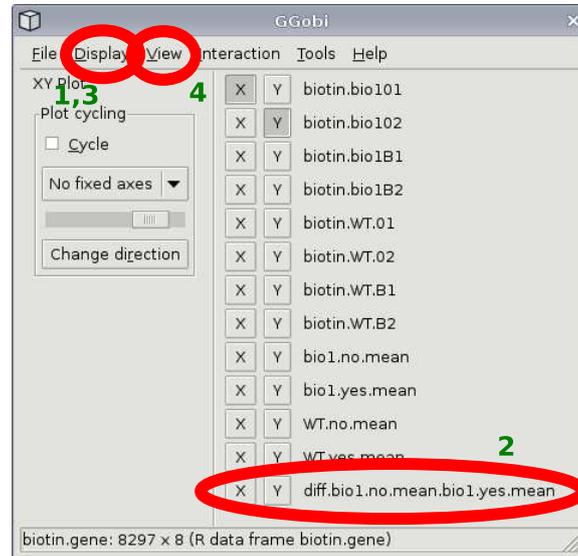
distribution of the difference.

Figure 10: The **GGobi** control panel, used for creating and configuring **GGobi** plots. The `File` menu supports loading and saving of **GGobi** datasets, but the **explorase** user is expected to load and save data through the **explorase** GUI. The `Displays` menu contains items for opening each type of display, such as scatterplots and parallel coordinate plots. The `View` menu allows toggling between display view modes, such as histogram vs. XY plot. The `Interaction` menu has an item for activating each interaction mode, such as the brush. The `Tools` menu contains various utilities. Below the menubar are two panes. The left contains options for the current interaction mode. The other lists the variables in the current dataset and has toggle buttons for specifying which variables are plotted in the current display. The numbered red circles drawn on top of the screenshot illustrate the steps to create the plots mentioned in the example.

To create **GGobi** plots follow these steps (as illustrated in Figure 10):

1. Use the **GGobi** control panel window (shown in Figure 10): select the `New Scatterplot Display` option from the `Displays` menu.

2. Select the two mean variables by clicking on the `X` button next to the `bio1.no.mean` label and the `Y` button next to the `bio1.yes.mean` label.

3. Select the `New Scatterplot Display` option from the `Displays` menu.

4. To change the scatterplot to an ASH plot (histogram), select the `1D Plot` from the `View` menu.

5. Click the `X` button next to the `diff.bio1...` variable, so that the histogram shows the distribution of the differences.

One way to verify that those genes are indeed dependent on biotin treatment would be to fit linear models using **limma**, including effects for the genotype, biotin treatment, and their interaction.
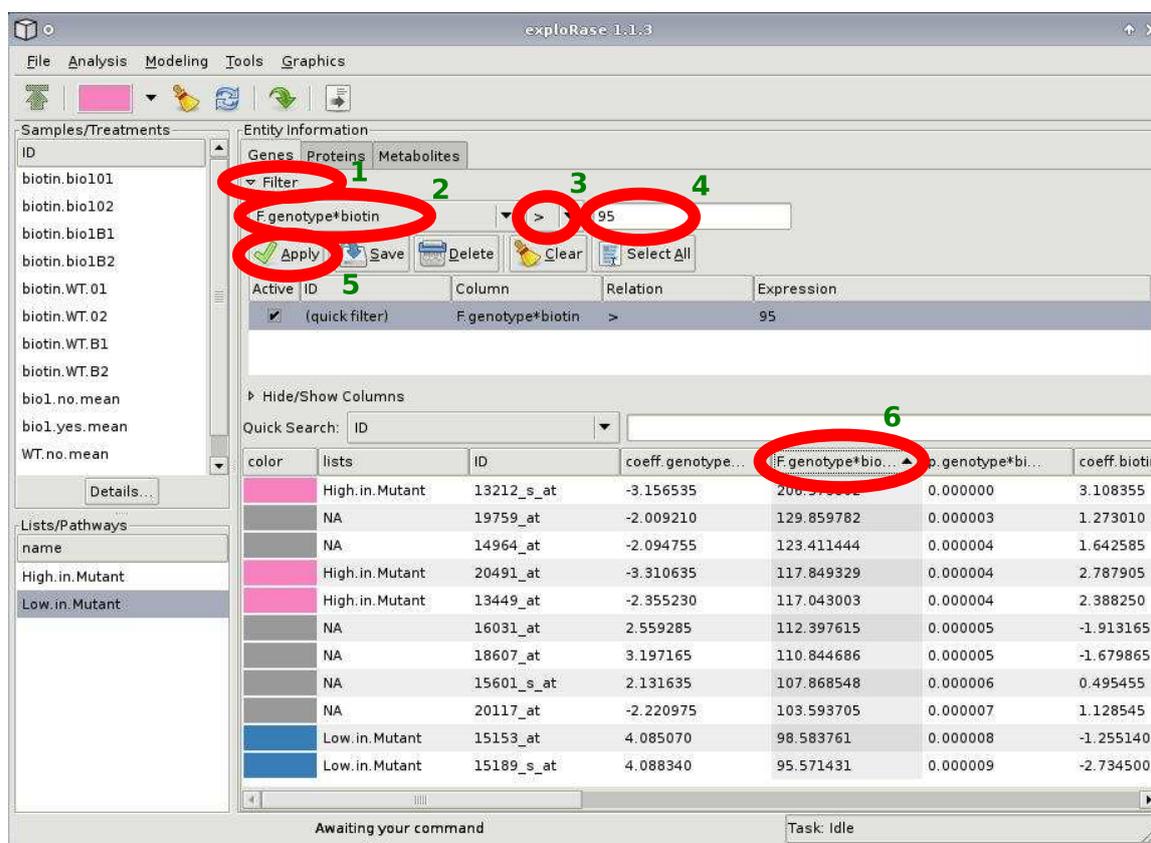
Figure 11: **limma** results for the biotin data. Only the entities with an F statistic > 95 for the interaction of genotype and biotin are displayed. The numbered red circles drawn on top of the screenshot illustrate the steps to filter the table.

To do this, use the **limma** interface in **explorase**:

1. Choose the `Linear modeling (limma)` option from the `Modeling` menu (in the menubar in Figure 1). This should open the `Linear modeling via Limma` dialog (shown on the left in Figure 7).

2. From the list of factors in the `Linear modeling via Limma` dialog, select the checkbox for the interaction of genotype and biotin. Note that this automatically selects the individual factors.

3. Click the `Apply` button to run **limma**.

Figure 11 shows the F values for the interaction of biotin and genotype. The table is sorted by the F value and filtered so that only the genes with the largest F values ($F > 95$) are included in the table. As one might expect, several of the pink- and blue-colored genes have extreme F values, indicating that biotin treatment has a genotype-dependent effect on those genes.

To identify those genes with the largest F values, follow these steps (as illustrated in Figure 11):

1. Click on the `Filter` label above the entity information table, so that the filter GUI is shown.

2. Select `F.genotype*biotin` from the left-most drop-down menu in the filter panel.

3. Change the second combo box to `>`.

4. Enter "95" into the text field to the right.

5. Click `Apply` to apply the filter rule (F.genotype*biotin > 95). Genes with an *F* value less than 95 are now excluded from the entity table.

6. Click the header of the `F.genotype*biotin` column to sort by it.

One outlier is easy to recognize even from the table: 13212_s_at. The annotations in the table describe 13212_s_at as a glycosyl hydrolase. The functions of the other outlying genes, if known, may be found in the table, and if more information is needed, clicking the `AtGeneSearch` button in the toolbar spawns a web browser and queries the **MetNetDB** (Wurtele *et al.* 2003) for additional details.

This analysis could continue along many paths. For example, the user might search for genes that are similar to the 13212_s_at using the distance measures in **explorase** (from the `Analysis` menu), or could continue to inspect the output of **limma** using **GGobi** graphics. Lists of genes could be exported to **MetNet** for pathway display (MetNet/Cytoscape) or evaluation in the context of public microarray data (MetaOmGraph) (Wurtele *et al.* 2007). This example demonstrates only a fraction of the potential of **explorase**.

7. Technical considerations

7.1. Suggested limit on number of samples

There are several reasons behind the suggestion that the number of samples (columns) in the experimental data be limited to a relatively small number (approximately 50 in our experience).

- The **explorase** and **GGobi** GUIs are not designed for a large number of variables. In particular, it is cumbersome to navigate through and operate on the variable selection panels.

- The analytical methods in **explorase** have not been designed/selected with a large number of samples in mind, though many of the same methods would still apply.

- The **explorase** implementation (and, in general, R itself) has not been heavily optimized (in terms of space and time) for large numbers of samples nor extremely large data matrices.

The **explorase** package could be modified to handle experiments with large numbers of samples, but it may be more feasible to develop a separate tool for that purpose.

7.2. Software infrastructure

explorase is written in the R language, facilitating integration with R analysis packages. This also enables other R packages to integrate with **explorase** via its public API that is documented through the online R help in the **explorase** package.

In order to provide its GUI, **explorase** relies on the **RGtk2** package (Lawrence and Temple Lang 2008), a bridge from R to the **GTK+** 2.0 cross-platform widget library (The **GTK+** Team 2008). **RGtk2** allows **explorase** to present, completely from within R, a visually pleasing, featureful GUI that is identical across all major computing platforms.

GGobi serves as the visualization component of **explorase**. The **rggobi** package (Temple Lang and Swayne 2001; Temple Lang *et al.* 2008) links R with **GGobi**. With **rggobi**, R packages are able to load data from R into **GGobi**, retrieve **GGobi** datasets into R, get and set the color of observations, create and configure displays, and more. **explorase** uses **rggobi** to load high-throughput datasets and synchronize the color of observations in **GGobi** plots with the colors in the entity information table in the **explorase** GUI. This provides the key visual link between the GUI of **explorase** and the visualizations of **GGobi**.

8. Related work

explorase is unique among open-source tools in its integration of interactive graphics with R statistical analysis beneath a GUI designed especially for the biologist. The commercial microarray analysis program **GeneSpring** links to R and **Bioconductor** but offers limited interactive graphics. The free program **Cytoscape** (Shannon *et al.* 2003) is designed for viewing and analyzing experimental data in the context of biological networks and is integrated with R via plugins. However, it lacks interactive graphics outside of its network diagrams.

Many GUIs have been constructed in R, including several in **Bioconductor**. The **limmaGUI** package (Smyth 2005) provides a GUI that leads the user from preprocessing microarray data to modeling it with **limma** and producing reports. Unfortunately, **limmaGUI** lacks the interactive graphics and breadth of analysis features of **explorase**. The **Bioconductor iSPlot** (Whalen 2005) package provides general interactive graphics using the R graphics engine but offers only a small subset of **GGobi**'s functionality. **Rattle** (Williams 2008) is an **RGtk2**-based GUI that leverages R as it guides the user through a wide range of data mining tasks.

9. Conclusion

The **explorase** package is an effective tool for analyzing and visualizing high-throughput biological data. Its direct access to R analysis packages, such as **limma** and others from the **Bioconductor** project, allow it to take advantage of the latest advances in statistical methods for bioinformatics. The integration with **GGobi**, including synchronized brushing and the ability to add analysis results as **GGobi** variables, empowers **explorase** to display a wide range of interactive multivariate graphics. All of these advanced statistical features are enveloped within a simplified GUI that is tuned for a biologist.

explorase has not yet realized its full potential. There are three predominant directions of planned improvement. The network visualizations in **GGobi** will be integrated with a convenient and reliable means for matching the identifiers of experimental measurements and

nodes in biological networks. This will automatically match identifiers and provide diagnostics to help the biologist ensure the fidelity of the matchings. Analysis methods will be expanded, with a particular focus on clustering and metabolomic analysis. Finally, the visualization of categorical data, such as GO terms and cluster assignments, will be enhanced.

Acknowledgments

References

Cook D, Hofmann H, Lee EK, Yang H, Nikolau B, Wurtele E (2007). "Exploring Gene Expression Data, Using Plots." *Journal of Data Science*, **5**, 151–182.

Friedl J (2006). *Mastering Regular Expressions*. O'Reilly, 3 edition.

Gentleman RC, Carey VJ, Bates DM, Bolstad B, Dettling M, Dudoit S, Ellis B, Gautier L, Ge Y, Gentry J, Hornik K, Hothorn T, Huber W, Iacus S, Irizarry R, Leisch F, Li C, Maechler M, Rossini AJ, Sawitzki G, Smith C, Smyth G, Tierney L, Yang JYH, Zhang J (2004). "**Bioconductor**: Open Software Development for Computational Biology and Bioinformatics." *Genome Biology*, **5**, R80. URL http://genomebiology.com/2004/5/10/R80.

Johnson RA, Wichern DW (2002). *Applied Multivariate Statistical Analysis (5th ed)*. Prentice-Hall, Englewood Cliffs, NJ.

Lawrence M (2007). "**explorase** Website." URL http://www.metnetdb.org/MetNet_exploRase.htm.

Lawrence M, Temple Lang D (2008). ***RGtk2**: R Bindings for Gtk 2.8.0 and Above*. R package version 2.12.5, URL http://CRAN.R-project.org/package=RGtk2.

R Development Core Team (2008). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL http://www.R-project.org/.

Shannon P, Markiel A, Ozier O, Baliga NS, Wang JT, Ramage D, Amin N, Schwikowski B, Ideker T (2003). "**Cytoscape**: A Software Environment for Integrated Models of Biomolecular Interaction Networks." *Genome Research*, **13**(11), 2498–2504. doi:10.1101/gr.1239303. http://www.genome.org/cgi/reprint/13/11/2498.pdf, URL http://www.genome.org/cgi/content/abstract/13/11/2498.

Smyth GK (2005). "**limma**: Linear Models for Microarray Data." In R Gentleman, V Carey, S Dudoit, R Irizarry, W Huber (eds.), "Bioinformatics and Computational Biology Solutions Using R and **Bioconductor**," pp. 397–420. Springer-Verlag.

Swayne DF, Temple Lang D, Buja A, Cook D (2003). "**GGobi**: Evolving from **XGobi** into an Extensible Framework for Interactive Data Visualization." *Computational Statistics & Data Analysis*, **43**, 423–444.

Temple Lang D, Swayne D, Wickham H, Lawrence M (2008). *rggobi: Interface Between R and GGobi*. R package version 2.1.8-1, URL http://CRAN.R-project.org/package=rggobi.

Temple Lang D, Swayne DF (2001). "**GGobi** Meets R: An Extensible Environment for Interactive Dynamic Data Visualization." In K Hornik, F Leisch (eds.), "Proceedings of the 2nd International Workshop on Distributed Statistical Computing, March 15–17, 2001, Technische Universität Wien, Vienna, Austria," ISSN 1609-395X, URL http://www.ci.tuwien.ac.at/Conferences/DSC-2001/Proceedings/.

The **GTK+** Team (2008). "**GTK+**: The GIMP Tool Kit." URL http://www.gtk.org/.

Whalen E (2005). *iSPlot: Linking Plots*. R package version 1.12.0, URL http://bioconductor.org/packages/2.1/bioc/html/iSPlot.html.

Williams G (2008). *rattle: GNOME R Data Mining*. R package version 2.3.1, URL http://CRAN.R-project.org/package=rattle.

Wurtele E, Li J, Diao L, Zhang H, Foster C, Fatland B, Dickerson J, Brown A, Cox Z, Cook D, Lee E, Hofmann H (2003). "**MetNet**: Software to Build and Model the Biogenetic Lattice of Arabidopsis." *Computational Functional Genomics*, **4**, 239–245.

Wurtele E, Ling L, Berleant D, Cook D, Dickerson J, Ding J, Hofmann H, Lawrence M, Lee E, Li J, Mentzen W, Miller L, Nikolau B, Ransom N, Wang Y (2007). *Concepts in Plant Metabolomics*, chapter **MetNet**: Systems Biology Software for Arabidopsis. Springer-Verlag.

Affiliation:

Michael Lawrence
Program in Computational Biology
Division of Public Health Sciences
Fred Hutchinson Cancer Research Center
1100 Fairview Ave. N.
PO Box 19024
Seattle, Washington 98109-1024, United States of America
E-mail: mflawren@fhcrc.org