Guide to Microarray Analysis

Damian Verdnik, Ph.D.

Last updated: January 14, 2004

1 Principles

The mathematical techniques of microarray analysis have been known to strike fear into the hearts of experienced biologists. The purpose of this guide is to demystify microarray analysis so that biologists can reclaim their data from the statisticians and computer scientists who are taking over their discipline.

Despite what your more mathematically inclined colleagues may tell you, microarray data is relatively straightforward. It does not require any new statistical theories for its analysis. Once you have analyzed your microarray images in GenePix Pro Software and imported the GenePix Results (GPR) files into the Acuity database, there is a relatively small number of filters and transformations that you need to apply to it before you can start extracting meaningful biological information from it.

Biologists have been deluged in recent years by a large number of data mining techniques that purport to be able to discover all sorts of information that is hidden in microarray datasets. These techniques may be able to do all that is claimed for them. However, there is one thing that they cannot do, because it is the prerogative of the biologist alone. Because it is so important to keep in mind and apply at every stage of microarray analysis, let us enshrine it in a principle:

First principle of microarray analysis

The output of every analysis algorithm that has been applied to microarray data is meaningless until a biologist validates it.

You understand the biology of your own experiments better than any analysis algorithm. With a small amount of knowledge of the mechanics of microarray analysis, you can very quickly judge the performance of any analysis algorithm on your experimental data, and whether or not it is useful.

You can think of this first principle in another way: *garbage in, garbage out*. Analysis algorithms are only as good as the data that they use as their inputs.

For example, suppose you are doing a microarray experiment on different types of cancer, and you are looking for substances that discriminate between the types. However, the samples come from various different sources: some are peripheral blood, some are fresh tissue and some are frozen tissue. There will be some substances that behave differently among samples just because of the differences in preparation methods. These substances may even show a higher level of differential expression than any other substances in the dataset, so an algorithm looking for discriminating substances will pick them out as being the best discriminators of the cancer types. But these substances will only be artifacts of your sample preparation methods, and you as a biologist will be able to recognize this immediately. No matter how powerful it is, an analysis algorithm used unwisely will not be able to take into account these differences.

Having said that, you can only use your understanding of the biology to evaluate an algorithm's output if you have control of your data at every stage of the analysis process. This is also crucially important, so let us formulate a second principle:

Second principle of microarray analysis

Data and the transformations that are applied to them must be available and transparent at every stage of analysis.

One of the aims of this guide is to empower biologists to take back the analysis of their data from black box software. You cannot do this unless you have access to your data. For this reason, one of the design principles behind GenePix Pro and Acuity is to allow users access to all their data at all times.

2 Data Types

Once a microarray image is analyzed in GenePix Pro and saved as a GenePix Results (GPR) file, the GPR file is all that you have to reconstruct what happened to the samples on that microarray. Although Acuity imports analysis JPEG images that GenePix Pro saves during its analysis, and although you can refer back to them in Acuity's Features tab, they are used only for qualitative analysis; for example, to make sure that all the features affected by an artifact are flagged Bad. It is the GPR file that contains the details of the experiment. The entire purpose of a program like Acuity is to transform and display data so that the huge amount of information that exists in the set of GPR files from an experiment can be analyzed and interpreted.

With that in mind, it should be clear that every column of data from every GPR file is your friend. Every column says something about the image, and hence about your experiment.

To see a list of the various data types in a GPR file and available for analysis in Acuity, open the *Current Data Type to Retrieve* dialog box. The top pane lists every GPR data type (i.e. every column from the GenePix Pro Results tab), while the bottom pane lists the operations that can be performed on the values of replicate features as the data is retrieved from the database (more about this later in "Dataset Filtering and Management").

Even in a simple two-color experiment, GenePix Pro extracts approximately 50 different data types for each microarray. There are feature intensities, background intensities, ratio types, sums of various sorts, threshold parameters, and each of these has several different variations. To use all this data wisely, you need to know a few basic facts about microarray data types.

2.1 Means and medians

At every stage of microarray analysis, large numbers of data points are combined to produce single representative measures: pixel values are combined to produce spot intensities, intensities are combined to produce ratios, replicate ratios are combined to produce substance values. Each time this is done, you have the choice of using the mean or the median to produce a representative value.

The mean is sensitive to outliers in the set, and consequently a single very large or very small value in an otherwise uniform set of numbers can produce a mean value that is no longer representative of most of the members of the set.

The median is the number for which half the data points are higher and half are lower (or if there is an even number of members of the set, it is the mean of the two in the middle). A few large or small values in a set do not affect the median very much, as all the representative numbers fall in the middle of the sorted list. If you are looking for a value that is representative of most of the numbers in the set, use the median over the mean.

The advantage of having both the median and the mean is twofold: you have the choice of using one or the other, and you can look at the difference between the median and the mean as a way of detecting the presence of outliers.

Other statistical quantities commonly reported can also tell you about the variation in your data:

- The **standard deviation** can be thought of as the average distance that individual data points fall from the mean: the larger the standard deviation, the more widely spread are the values in the set.
- The **coefficient of variation** is the standard deviation divided by the mean, expressed as a percentage; that is, it is the standard deviation normalized by the mean. The coefficient of variation is a way of estimating the spread of values in a set independently of the scale of the values.

2.1.1 A note on calculating means

If you want to verify by hand any of the calculations done in Acuity (such as means), be aware that different data types must be handled differently. The mean of ratio values, for example, is calculated in log space (it is known as a geometric mean) where the ratios are normally distributed and additive (see the next section).

2.2 Ratios and logs

In standard two-color ratio experiments, we are interested in calculating a ratio value for each spot as a measure of the activity of a substance, so we commonly look for features with ratios above 2 or less than 0.5, for example, or above 4 or less than 0.25. However, typically we do not work with ratios, but with log ratios.

You can see why we do this with a very simple example. Consider two spots on a microarray, one with a ratio of 2, and the other with a ratio of 0.5: one is twofold induced, the other is twofold repressed. The mean of these two spots should therefore be 1. But the mean of 2 and 0.5 is 1.25. What has gone wrong?

The problem is that ratios are not normally distributed; whereas log ratios are. If we take logs (to the base 2) of 2 and 0.5, we get 1 and –1, and then the mean of these is 0. The antilog of 0 is 1, and we get the mean that we expect.

(Internally, Acuity keeps track of properties of each data type so that when performing various mathematical operations, such as averaging and normalization, data types are always handled appropriately. You can view and edit these properties in the *Remap Data Types* dialog box.)

2.2.1 Scatter plots

When dealing with entire microarrays, scatter plots and histograms can very quickly tell us a great deal about what is occurring on a microarray. They also demonstrate very vividly the value of using log ratios over ratios.

Suppose we are interested in whether or not the ratios on a microarray change with the intensity of features; that is, we want to know if there is a dye bias in our data. We might do a scatter plot of the sum of medians against the ratio of medians, as shown in Figure 1a.

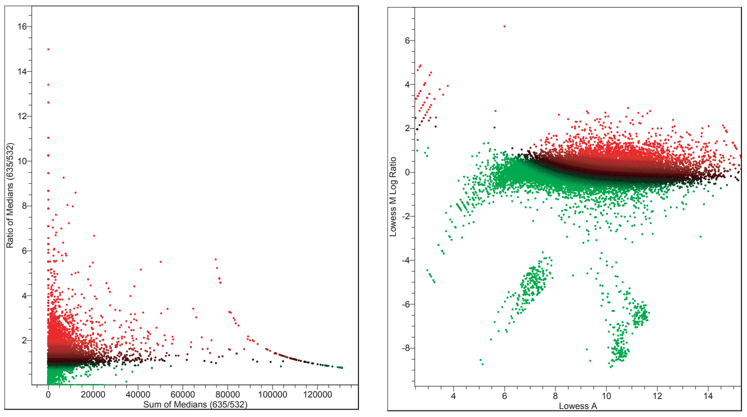


Figure 1a & 1b

This particular scatter plot fails in two ways. First, any substance that is repressed has a ratio between 0 and 1, whereas any substance that is induced has a ratio greater than 1. This means that the distribution of ratios on this scatter plot is very unequal: all repressed genes are squeezed into the interval between 0 and 1, while the induced genes range between 1 and infinity. It is very difficult to see what is happening to features with ratios less than 1.

The second failure is that a large number of substances are squeezed into the low-intensity part of the scatter plot, and it is difficult to differentiate what is happening in that part of the plot.

If, however, we display the same scatter plot with log ratio data on the Y-axis, and display sum of medians data on a log X-axis, the structure in the data is revealed (Figure 1b).

Note the advantages in this form of the scatter plot:

- On the Y-axis, the log ratio of data is displayed symmetrically about zero, so we can equally identify induced and repressed genes.
- On the X-axis, the sum of medians data plotted on a log axis gives prominence to the low intensity points in which we are interested.

Now that we have a scatter plot that we can use, we can discern at least three obvious trends in this data.

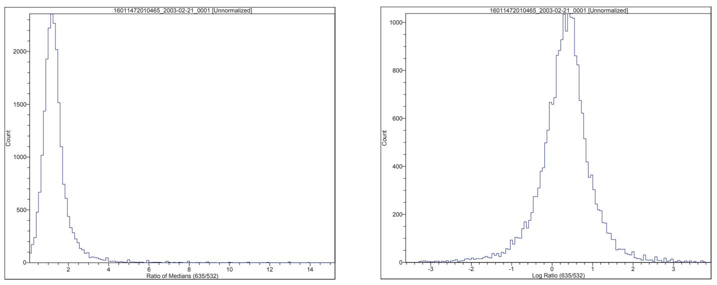


Figure 2a & 2b

First, we see that a majority of points falls along the log ratio equals zero line. These points correspond to substances with little or no differential expression.

Second, on the intensity axis, the structure around low-intensity features is more obvious. There is a trend in the data towards negative ratios at low intensities. This could mean that there is some systematic error in the array-production process.

Third, we see two clouds of points with large negative log ratios that are separated from the main distribution. These are two different populations of controls, which explains their tight distributions.

2.2.2 Histograms

We see similar differences if we do a histogram of the ratio data (Figure 2a).

A histogram of the ratio of medians (Figure 2a) is much less informative than a histogram of the log ratio data (Figure 2b) over the same data range, for the reasons mentioned above: when transformed into log ratios, ratios of less than 1 have a similar distribution to ratios greater than 1. We look at histograms in more detail in the next topic, Normalization.

The lesson for microarray data is simple: always use log ratio values (or plot ratio data on a log axis) if you are interested in the distribution of ratios across an entire array.

3 Normalization

Normalization is the process of adjusting experimental data so that:

1. Data from a single experiment are as accurate as possible. For example, if we are using an instrument that adds a well-characterized offset to our data values, then we correct for this offset.

2. Data from different experiments can be compared to each other. For example, we may wish to correct for variations in sample preparation between experiments by forcing all data to fit a specified distribution.

In microarray experiments, this typically involves adjusting the data on a single array, and then adjusting the data across arrays.

Normalization is also discussed in the Acuity printed manual.

3.1 Normalizing data on a single array

3.1.1 Linear Normalization

Data on a single microarray may need to be adjusted for a number of reasons. The most common and most easily understood reason is to correct globally unbalanced dye signals that can be caused, for example, by the scanner PMTs not being balanced. We want the PMTs to be set so that a feature that has two fluroescent lables bound to it with a 1:1 ratio actually produces a ratio of 1.0 when it is scanned. If the PMTs are set so that this is not the case, then the ratio value of every feature on the microarray will need to be adjusted.

This sort of global imbalance is corrected in the following way. In many whole-genome gene expression experiments we expect the mean of all ratio values to be close to 1.0 because for any given experimental system relatively few genes are differentially expressed. However if the microarray contains a small or functionally specific set of genes, we may expect many of them to be differentially expressed. In this case normalizing the data to force a ratio of 1.0 may mask important differential expression. As stated in the first principle above, as the biologist you must consider whether any calculation is appropriate for your experiment.

If the mean ratio is different from zero, then we can force it to be zero by adjusting every feature by constant multipliers, using the following method:

- Calculate the mean of the ratio of every feature (let us say that it is 1.21).
- Take the square root of this number (= 1.1).
- Multiply all numerator wavelength values (e.g. F635 Median) by 1/1.1 = 0.91.
- Multiply all denominator wavelength values (e.g. F532 Median) by 1.1.
- The new global ratio value is now 1.21 × (0.91/1.1) = 1.0.

0.91 and 1.1 are the normalization factors.

In Acuity, this sort of linear normalization is called Ratio-based normalization, and can be performed in the Normalization Wizard by the click of a button.

(Note that if you have already created datasets and analyzed data from a microarray, you cannot normalize the microarray in this way. This is because the existing analysis results rely on unnormalized data; if the data are normalized after the analyses, the analyses would be invalid. Therefore, you should always normalize microarrays before analyzing them. Typically, do it immediately after importing them.)

Outliers on a microarray can strongly skew the global mean ratio, so it is advisable to normalize on the mean ratio of features with ratios between 0.1 and 10, as this will better represent the global shift in ratios (Ratio-based normalization is also the normalization method implemented in GenePix Pro).

It is standard practice when analyzing microarrays to normalize them with ratio-based normalization.

A good way of understanding the effect of Ratio-based normalization on your microarray is to plot a histogram of the log ratio data in the Normalization Viewer. As you can see in Figure 3, the unnormalized distribution (Figure 3a) is symmetrical about a vertical axis but not centered on zero. Applying a ratio-based normalization shifts the distribution so that it is centered on zero (Figure 3b), but it does not change the distribution's shape, as every feature is shifted by the same factor.

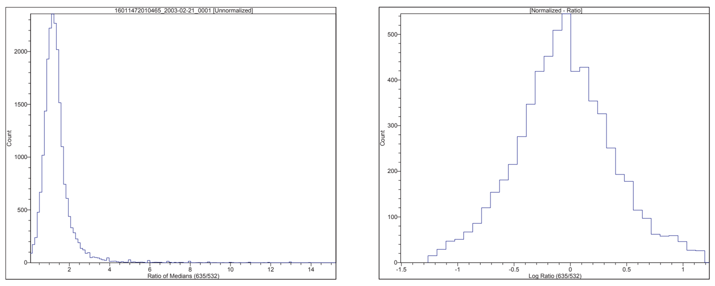


Figure 3a & 3b

3.1.2 Non-linear Normalization

It is important to understand exactly what ratio-based normalization can and cannot do. The Ratio-based normalization described above is excellent when the histogram of the log ratio is already symmetrical about a vertical axis, and it merely needs to be centered on zero. However, this is not always the case with microarray data. It is common to see histograms that are not symmetrical about a vertical axis, and such distributions are not corrected by Ratio-based normalization, as in Figure 4.

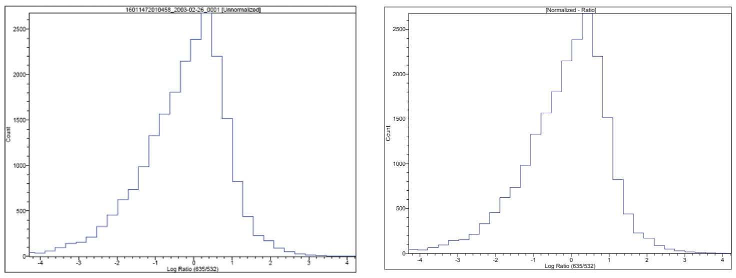


Figure 4a & 4b

In Ratio-based normalization, because we assume that the global distribution of ratio data on a microarray is symmetrical about zero (in log space), it should not be surprising that we cannot correct the distribution. However, there can be effects on microarrays that are intensity-dependent. It is common to see microarrays that have unbalanced ratios in just one part of the intensity spectrum.

In the histogram in Figure 4a, the global mean ratio is very close to 1, because the areas under the histogram on either side of the log ratio = 0 axis are equal. Ratio-based normalization barely affects the histogram. As you can see, when it is normalized the distribution is shifted very slightly to the right, but its shape does not change. If we want to change the distribution so that it is symmetrical about zero, we need to use a normalization method such as lowess normalization that separately normalizes sub-intervals of intensity.

In Figure 5, we show the effects of lowess-normalization on the microarray from Figure 4 by using the Acuity Normalization Wizard. Lowess normalization uses the assumption that log ratio data should be symmetrical about zero in any particular intensity interval. It changes the shape of the histogram quite dramatically, as you can see in Figure 5b. Not only is the distribution now symmetrical, but also the upper and lower ranges have been tightened so that the overall dynamic range is smaller.

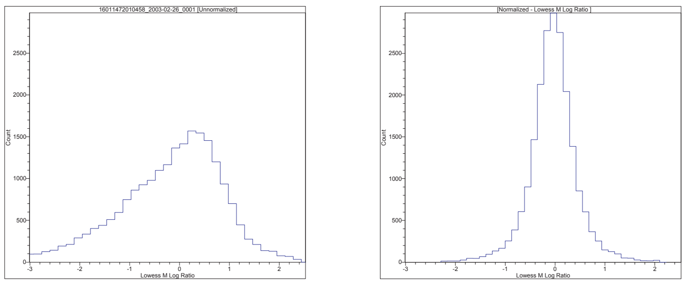


Figure 5a & 5b

Another way of displaying the effects of lowess normalization on a microarray is through what is called an M vs A plot.

On the X-axis we plot the A data type, which is a combined measure of intensity defined by:

$$0.5\log_2(\text{F635 Median} \times \text{F532 Median})$$

You can think of A as being similar to a logged sum of medians.

On the Y-axis is the log ratio, known as M, hence the name M vs A.

In the unnormalized scatter plot at the top of Figure 6a, you can clearly see that at low intensities (at the left-hand end of the X-axis), the log ratio values are skewed to the negative, whereas at high intensities they are skewed to the positive.

In the lowess-normalized scatter plot (Figure 6b), the distribution is now symmetrical about zero at all intensity values. (The small clouds of outliers at A = 6 and A = 10 are control spots that are supposed to have negative log ratio values.) At every intensity interval, the data have been redistributed symmetrically about the M = 0 axis.

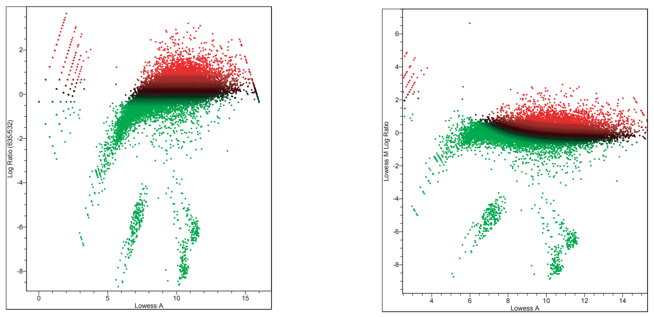


Figure 6a & 6b

So graphically we can see what is being done to data when it is lowess normalized, but what physical array effects are we correcting? There is no unequivocal answer to this question.

As argued further in the Acuity printed manual, you should be conscious of exactly what is being done to your data when you apply a lowess normalization, as the distribution can be changed quite radically. In particular, we do not recommend that you use lowess-normalized data unless you understand exactly why your data is distributed in the way that it is, and hence why it can be validly lowess-normalized.

Having said that, drawing M vs A plots and observing the effects of lowess normalization is an excellent diagnostic tool. On good arrays you should not see ratios vary strongly with intensity. If they do, there may be something systematically wrong with your array-production process.

One point to keep in mind is that many of the features at the far left of the M vs A plot that are most strongly shifted away from M = 0 are likely to be unreliable for other reasons. The curved part of the distribution in Figure 4 occurs mainly for A values less than 6, which might correspond to a sum of medians value between 100 and 200. Removing all those features makes the distribution look a lot better, and hence it may not be worth lowess normalizing at all.

3.2 Normalizing data across arrays

Once each microarray in a dataset is normalized individually, you may want to normalize them as a set, so that they share the same distribution, for example.

These sorts of transformations are described in the "Pre-Processing" section of the "Clustering" topic.

4 Dataset Filtering and Management

Datasets are the units of analysis in Acuity. Typically, a dataset consists of all the reliable data from a set of microarrays that together form an experiment.

There are two main reasons why we might create a dataset from only a subset of the available data, instead of from each feature from every microarray in an experiment:

1. We remove unreliable data from the dataset. For example, we remove data points derived from slide defects such as smears.

2. We remove uninteresting data from the dataset. For example, we may have control features used for normalization that are not needed for downstream analysis; or, we remove substances that do not show any interesting behavior in order to make the analysis task more tractable.

Removing unreliable data is one of the more treacherous tasks facing the microarray researcher, due to the subjective nature of what counts as "good" data, the variability in data quality across microarrays, the lack of accepted standards for good data, and the problem of translating image-based defects into numerical conditions on array data types.

Removing uninteresting data can be more or less controversial depending on what you regard as "uninteresting". That control spots are not required for downstream advanced analysis may seem self-evident, but it is a questionable assumption, as control spots by their nature should cluster together and so can be used to validate advanced analyses. Other dataset filters that claim to remove uninteresting data points, such as fold-change filters, may excise potentially important data points along with the uninteresting data, so they should be used with caution (more about this in section 4.4.2 and section 6).

4.1 Measures of feature quality

One of the challenges to both novices and experts in microarray analysis is how to translate the quality judgements that we make confidently by eye when looking at microarray images, into a formalism that can be applied reproducibly on a large number of microarrays.

The easiest way to do this, and it is relatively easy, is to make a list of common feature and slide defects, and then translate them into numerical conditions on GenePix Pro and Acuity data types. Note that all these conditions should be applied to microarrays that have already been normalized.

Here is a non-exhaustive list of common defects, in no particular order:

1. Feature is smeared into a neighboring feature;
2. Feature is very close to background;
3. Feature has a hair or a scratch through it;
4. Feature is in pieces;
5. Feature is saturated;
6. Feature pixels have highly non-uniform intensities;
7. Feature has a highly non-uniform background.

Notice that each of these defects is evaluated for each feature individually.

Condition 1 almost always results in a feature being Not Found by the GenePix Pro spot-finding algorithm, so we should exclude Not Found flagged features, as well as Bad and Empty features:

Flags >= 0

This defect also leads to a high background, which is identified by other conditions.

Condition 2 can be quantified in a number of different ways. One very powerful way is to demand that the signal-to-noise ratio is above the detection limit of three in both channels:

SNR 635 > 3 AND SNR 532 > 3

Alternatively, if you know the median background level on your slides, you can create a condition based on the absolute intensity in each feature:

Sum of Medians (635/532) > 200

You can change the value of 200 to suit the background level on your arrays.

A third alternative is to place a condition on the percentage of pixels two standard deviations above background:

% > B635+2SD > 55 AND % > B532+2SD > 55

Again, the value 55 can be adjusted to suit your arrays.

Conditions 3 and 4 can be formalized by using the Circularity metric introduced in GenePix Pro 5.0 (assuming that you have used non-circular feature-indicators):

Circularity > 80

Condition 5 is relatively simple to quantify:

F635 % Sat. < 2 AND F532 % Sat. < 2

Conditions 6 and 7 can also be quantified in a number of ways. For the background intensity one can place a condition on the coefficient of variation of the local background in each channel:

B635 CV < 25 AND B532 CV < 25

The coefficient of variation is not entirely reliable as a measure of uniformity, because when the mean is small, the CV can be large even for a uniform pixel distribution.

We could do the same for the feature intensities, but for instructional purposes let us use a condition on the Rgn R2 value, which measures the uniformity of pixel intensities in the region of the feature:

Rgn R2 > 0.6

With all these conditions in place, all we need to do is concatenate them with AND operators, and enter them into the Query Wizard. The end result of a query from the Query Wizard is an Acuity dataset.

Note that many of the thresholds in the above conditions are arbitrary. Whether you set a threshold of 25 or 30 for background CV, for example, is entirely up to you.

From long experience working with microarrays of many different types, we have found that the single most powerful measure of feature quality is the Rgn R2. Many labs use it, and depending on the quality of their arrays they set the threshold lower or higher: some use 0.5 or 0.4, while others use 0.7 or 0.75.

4.2 Measures of substance quality

It is worth emphasizing that each of the quality control conditions formulated above is evaluated for individual spots on single slides. The dataset may be created from hundreds of microarrays in a single experiment, but the quality control conditions are evaluated spot by spot.

The next level of filtering that we can apply is to evaluate data quality by comparing the value of replicate features on the same microarray, or on replicate microarrays. This is termed "substance" data quality, because in Acuity we average replicate features on each microarray to produce a representative value for each substance on the array.

4.2.1 Replication

Replication is the key to truly accurate data in any scientific experiment. When it comes to microarrays, scientists sometimes get confused about replication: what counts as replication, and how much is required. The rules for microarrays are no different than the rules for any other experimental technique:

- The only way to be truly confident in the results of any experiment is to perform the entire experiment a number of times.
- When repeating an experiment, vary apparently unrelated conditions as much as possible, such as reagents, slide types, and equipment such as hybridization chambers.
- In a single experiment it is always better to replicate across microarrays than within microarrays, because there is more chance of noticing array-specific biases across microarrays.

Whether you replicate within or across microarrays, it is only through replication that you can assess the variability in your data.

4.2.2 Replicate features on microarrays

To assess the quality of replicate features on a single microarray or replicate microarrays, you can use Acuity to calculate statistics such as standard deviations and coefficients of variation on replicate features.

Two replicates is the minimum number from which we can obtain any information at all; if the ratio values disagree, we can examine them on the image on the Features tab to see if both spots are reliable.

With three replicates, there will be cases where one is inconsistent, and therefore we can remove one of the features from downstream analysis. In the case above, we can with a reasonable amount of confidence say that there is an outlier.

As noted above when using CV to measure the uniformity of background intensity, CV is not reliable when the mean is small. It consequently not reliable on log ratio data, where most means are close to zero.

4.3 Replicate microarrays

Many researchers do replicate microarrays during assay development, or of crucial time points (e.g. time-point zero) to ensure that they have a reliable reference sample. Here we present a simple but powerful example of assessing the quality of three or more replicate microarrays using Self-Organizing Maps (SOMs):

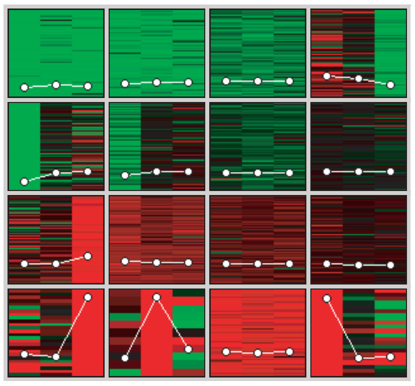


Figure 7

The SOM in Figure 7 is of three replicate ratio-normalized microarrays from the time-point zero of a large-scale experiment, where each microarray has around 6800 features. Each square in the SOM is a cluster of substances and the trace is the average profile of all substances in the cluster. The colors in each cluster are miniatures of the colors assigned to each substance profile.

On ideal microarrays, we would expect the values for each substance to be the same on every microarray, and hence we would expect each cluster to be a single color. Because microarray data can be noisy, on good-quality arrays we nevertheless expect a small amount of variation from array to array.

In the example above, we have a number of clusters with substances behaving as expected on good arrays, but other clusters, such as three of the four along the bottom, where the values are highly variable from one array to another. We can create a new dataset that excludes these clusters. Once this is done, we can run the SOM again.

This is a slightly cleaner dataset, but there is still a large amount of variability across the three microarrays.

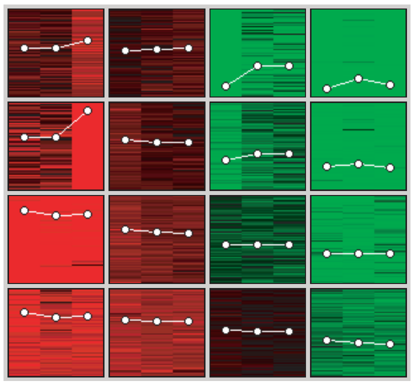


Figure 8

We expect every cluster to have a single color, because the ratio values should be the same on each microarray. In log space, we can formalize this condition by saying that we want each substance to have the same sign log ratio across the three microarrays.

This dataset (see Figure 8) contains around 5200 of the original 6800 substances, so we have removed around 25 per cent of the original data. However, we now have a dataset of replicate arrays where the replicates are all relatively close in value. The data filtering is not complete, but it has advanced a long way very quickly.

The great advantage of using a SOM is that it provides a very graphic method of quickly evaluating the quality of replicate arrays. The fact that clusters should be all red or all green when clustering replicate arrays means that a visual inspection method can be powerful. We can see this in another example.

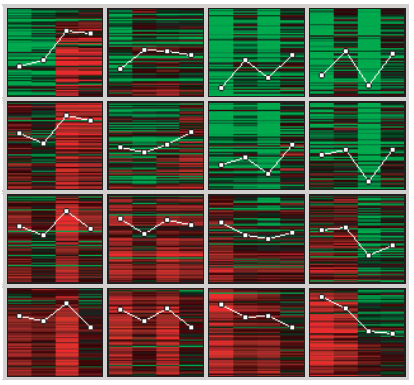


Figure 9

In the following SOM we have clustered four replicate arrays, where the second two arrays are dye swaps of the first two arrays:

The top left and bottom right clusters contain data that we expect to see in a replicate dye swap experiment, as the dye swaps are obvious from the first two to the second two arrays. In many other clusters, however, there is a disturbing inconsistency in the data. One can see this immediately from the SOM without doing any sophisticated statistical analysis.

See below for more on managing replicate microarrays.

4.4 Conditions across microarrays

The method described above as part of the Self-Organizing Maps technique of finding replicate substances with the same sign is a special case of filtering data by imposing conditions across more than one microarray. Once you have a dataset consisting entirely of good-quality features, you may wish to filter it further by imposing conditions based on substance behavior across all the arrays in the experiment.

4.4.1 Missing values

Having applied spot quality control criteria in the Query Wizard (described in 4.1) you may find that many substances in the dataset have missing values. That is, the substance is in the dataset because features from some microarrays passed the query criteria, but on other microarrays the features failed the criteria and hence are missing. Wherever a failure occurred, the cell in the main data pane in Acuity reports "<no data>."

Where a substance has a large number of missing values in a dataset, any clustering results containing that substance will have little meaning, so it makes sense to remove those substances before any downstream analysis.

Alternatively, if you want to eke as much out of your arrays as possible, you can leave missing values in a dataset. Hierarchical clustering is relatively robust with datasets containing missing values, and the other clustering algorithms have internal methods for dealing with missing values that you can specify in the cluster configuration dialog boxes.

In Acuity, you can always retrieve missing values from the database with the *Fill Empty Cells* command.

4.4.2 Fold-change conditions

When working with very large datasets, it is common to reduce the size of the dataset by removing substances that do not show some specified change in expression on some number of microarrays. This is called a fold-change filter. It is typically done for a number of reasons:

- Datasets with a very large number of substances take longer to analyze, and are awkward to manage.
- Many substances do not change expression in an experiment. It is assumed that if a substance does not show some minimal change in expression, then it is not going to be of biological interest.

For example, in a dataset consisting of ten microarrays, you may want to look at those substances with ratios greater than 2 or less than 0.5 on at least 2 microarrays. If using log ratio data, we can find all those substances with an absolute value greater than 1 on at least 2 microarrays. These two conditions are similar but not equivalent: the first finds substances that are up on two arrays or down on two arrays, whereas the second finds substances that are up on two, down on two, or up on one and down on one.

Use fold-change filters with care. The substances showing the largest changes in expression are not always those that are most biologically significant.

Further, one of the great strengths of microarrays is that we can do whole-genome experiments. A fold-change filter immediately introduces a bias to the analysis: it eliminates substances that do not pass an arbitrary fold-change cutoff. If we do not analyze all the data that we have, then we are not using microarrays to their full potential.

See Advanced Dataset Analysis for more on data filters.

4.5 Managing replicate microarrays

Once you have a dataset that has been filtered and cleaned of unreliable data points, you need to decide how to manage replicate microarrays in downstream analyses such as clustering. You have two choices:

1. Keep each microarray as a separate entity. This has the advantage allowing you to observe the differences among replicates. If the replicate arrays do not cluster together, for example, then you know that the variation among supposedly identical arrays is greater than the variation among different samples, and therefore that you may need to do your experiment again, or re-design it.
2. Combine each set of replicate arrays, for example by averaging. This has the advantage of smoothing out the variation in measurements on individual arrays, and providing a representative measure of expression across multiple arrays.

5 Clustering

5.1 Preconceptions

Biologists who are new to microarray analysis and data mining often share a number of preconceptions about clustering. Before we can discuss clustering in any detail, we need to explode these preconceptions.

Preconception 1: Once I have clustered my data, my experiment is finished

Clustering is not the end of your experiment. Clustering is an exploratory technique that you can use to investigate your data in many different ways, depending on the questions in which you are interested. If your data is at all good, clustering will raise more questions than it answers.

Before you come to clustering, you follow a relatively linear path in Acuity: import data, normalize, and filter datasets. Once you get to the point of clustering your data, the simple linear path stops. At the end of the path is a playground. There is no single way to use a playground. You just go ahead and play on it.

Clustering will answer simple questions, but it will raise more, many of them without simple answers.

Replace preconception 1 with: clustering will answer some questions, and raise many more. I must be prepared to spend a lot of time exploring my data.

Preconception 2: Before I can use clustering I need to know the best method to use on my data

There is no one clustering method that is best suited to your data. Software like Acuity provides many different methods of analysis because different methods provide different perspectives on your data.

Try them all. Experiment. Over time you will find that some methods are more informative than others, and you will use those.

Robust clusters in your data will be found by any clustering method. If quite different clustering methods produce different results, then perhaps the data do not fall into very strong groups.

Replace preconception 2 with: There is no best clustering method; there are only more informative and less informative methods.

Preconception 3: Clustering will find the real structure in my data

Clustering will find structure in your dataset. Given a dataset consisting of randomly generated microarray-like data, clustering will find structure. That is what clustering does. It does not mean that the structure has any basis in reality.

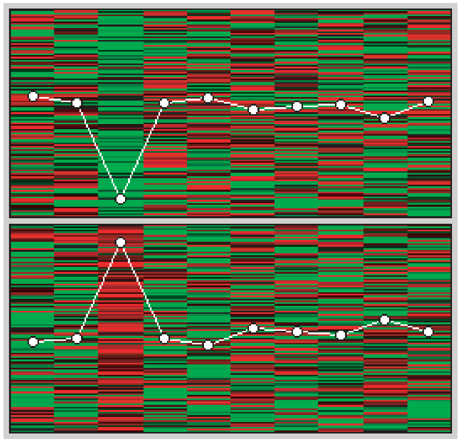


Figure 10

One way that graphically demonstrates this point is simply to analyze a random dataset. We generated a dataset consisting of 10 microarrays of 5000 substances each, and ran a Gap Statistic analysis, which estimates the optimal number of clusters to find in a dataset.

The Gap Statistic analysis suggested two clusters, so we performed a 1 $\times$ 2 SOM. The result, shown in Figure 10, demonstrates that there are indeed two clusters in the data: there is a clear partition based on the value on the third microarray.

What does this tell us?

First, we must be able to determine if the structure found by a clustering algorithm is more than what we would expect in random data.

Second, and more generally, the results of clustering need interpretation just as the results of any mathematical operation need interpretation. Even if the structure that the algorithm finds were more than one would expect from random data, one still must determine if it is biologically relevant.

Being a trained biologist, you must look critically at the results of clustering operations and decide if they are artifacts of sample preparation methods, for example, or if they represent true biological variation.

Replace preconception 3 with: Only the intelligence of a trained scientist can decide if the structure found by the clustering algorithm is biologically relevant and informative, or if it is an artifact of the experimental design, or the algorithm.

5.2 A classification of clustering methods

There are basically two classes of clustering methods in Acuity:

Hierarchical: every substance is related in a hierarchy to every other substance.

Non-hierarchical (K-Means, K-Medians, Self-Organizing Maps, Gene Shaving): substances are put in clusters, but the clusters have little or no relationship with each other.

(Principal Components Analysis (PCA) is organized under the Clustering menu in Acuity, but it is not really a form of clustering; more of it later.)

While this is a well-founded division of the clustering methods, it is not immediately informative. We have to tease out the implications of the division in order to see its consequences.

First, a major difference between the methods as organized in this way is how they are visualized. Hierarchical clustering requires a hierarchical visualization, and one way of doing this is with the familiar dendrogram. By contrast, when visualizing non-hierarchical clusters, each cluster can be displayed separately.

This is not a trivial difference. Visualizations are an extremely powerful method of displaying a large amount of data all at once in a way that is intelligible. Think of it this way: the output of a clustering algorithm is merely a table of numbers reporting the similarity of substances to each other. If Acuity only reported tables of numbers as outputs of the clustering algorithms, it would be useless. The power of a program like Acuity is in the way it makes analyzed data immediately interpretable, and it does that through its visualizations.

Through their different visualizations, the two classes of clustering methods will make you think about your data differently, so when analyzing a dataset it is always worthwhile doing at least one of each class.

Second, the methods differ in the inputs that they require, which also has consequences for how you think about your data. The main difference in this regard is that hierarchical clustering clusters the whole dataset as one cluster with internal structure, while the non-hierarchical methods require that you specify the number of clusters to find.

The requirement that you specify the number of clusters to find before you have clustered your data can cause confusion for newcomers to microarray analysis. (It is also the reason why Acuity includes the Gap Statistic algorithm, which provides an estimate of the optimum number of clusters in your data when using K-Means, K-Medians, or Self-Organizing Maps.) The confusion arises as a consequence of Preconceptions 2 and 3 above, that the point of clustering is to find the real structure in your data, and that there is a best method. Hierarchical clustering supports these preconceptions, because it seems to present a single unambiguous segmentation of your data.

The fact that non-hierarchical methods are entirely agnostic about the number of clusters in the data should help to dispel the preconceptions, but it should also make you look at hierarchical clustering differently. Hierarchical clustering is just as arbitrary as non-hierarchical clustering (as one must specify a similarity metric and a linkage method, for example), but it appears to be less arbitrary, because once the inputs are specified it provides a single segmentation of the data.

5.3 Pre-processing

Apart from the explicit normalization methods that transform your microarray data and whose results you can view in the Acuity interface, the Acuity clustering algorithms have a number of internal transformations that can be applied to data before it is clustered.

5.3.1 Row Centering

Row centering subtracts the row mean from each row. This transformation makes sense when we are interested in the shape of a response, and not its absolute value.

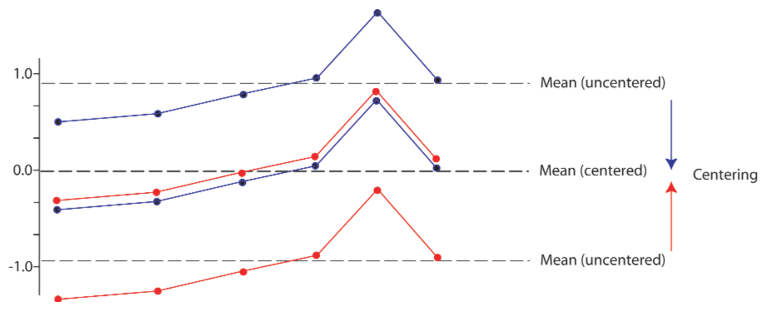


Figure 11

In Figure 11, we show the behavior of substances over time or across samples: when the profiles are mean centered, the algorithm ignores the absolute values and compares their profiles only.

Whether or not you use row centering will depend on whether or not you think the substances represented by the red and the blue traces above should be grouped together.

When we cluster the same dataset with and without centering, we see less diversity in the result that is row centered, because row centering removes a large amount of variation from a dataset. The simplest way to verify this for yourself is to run a Gap Statistic analysis without row centering, and then with row centering. You will find that the Gap Statistic without row centering predicts a larger number of clusters than the Gap Statistic with row centering. Further, the gap values for each cluster solution within each Gap Statistic analysis show more variation for the uncentered data than the centered data.

5.3.2 Row Scaling

Row scaling scales all rows so that they have the same range of expression (the same variance). This transformation makes sense when we are interested in the shape of a response, and not its magnitude. That is, once rows are scaled, substances are clustered together based on their expression profiles but independently of how strong their responses are.

In Figure 12, we show the behavior of substances over time or across samples: when the profiles are mean scaled, the algorithm ignores the absolute values and compares their profiles only.

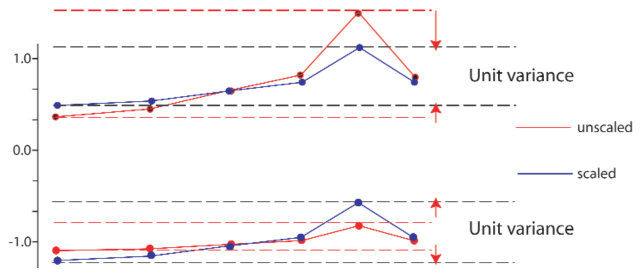


Figure 12

Whether or not you use row scaling will depend on whether or not you think the substances represented by the red traces above should be grouped together.

As with row centering, row scaling reduces the amount of variation in a dataset, which are reflected in the results of Gap Statistic analyses with and without row scaling.

5.3.3 Array Scaling

Array scaling scales the variance of each array in a dataset so that they have the same range of expression. This transformation makes sense when we are comparing arrays from different samples that have very different expression levels, such as tissue types, but we want the contributions from each array to be equal. That is, it is a way of transforming all arrays so that they have a common data range.

5.4 Hierarchical clustering

Hierarchical clustering consists of two separate mathematical operations:

- A similarity metric, which determines the similarity of each substance to every other substance.
- A linkage method, which is used to order the substances and construct the branches of the dendrogram.

When performing hierarchical clustering, you can choose to cluster substances, or microarrays, or both. You cluster microarrays when they do not already have some intrinsic order (as they do in a time course, when they are ordered by time, or some other variable dependent on time). However, it can also be instructive to cluster microarrays in a time-course experiment to see if in fact the time course is reproduced by the clustering algorithm. If it is not, then the data may not be very representative of the time course (perhaps because the sampling intervals are too far apart).

We can understand the output of the hierarchical clustering algorithm by examining how it is represented by its visualization (Figure 13a).

The most obvious thing you notice about the visualization is that there are two parts to it: a color table, and a dendrogram. The dendrogram is the part that is actually produced by the algorithm; the color table is an aid to analysis, but is merely the color profile of each substance as ordered by the algorithm.

The second thing to notice is the correlation scale at the bottom of the dendrogram. The algorithm calculates correlation coefficients, where 1 is perfectly correlated, -1 is perfectly anticorrelated, and 0 is uncorrelated (if you use a distance metric such as Euclidean distance instead of a correlation coefficient, the distances are transformed and scaled to the range –1, 1).

5.4.1 Metrics

Acuity includes a large number of similarity metrics, which are formally described in the printed manual.

The large majority of published microarray experiments that report hierarchical clustering results have used the Pearson Correlation as the similarity metric. Spearman and Kendall are very similar to Pearson, except that they operate on rank orderings, and so are less sensitive to outliers than the Pearson correlation.

The binary matching metrics are unsuitable for gene expression data where one has continuously varying expression levels, or any other dataset with continuous values. They are appropriate for experiments such as Comparative Genomic Hybridization, where one is looking for the presence or absence of some property.

5.4.2 Linkage

The different linkage methods typically produce orderings of substances that are very similar, which is shown by comparing color tables derived from the different methods, but they produce very different trees. Given similar color tables (i.e. substance orderings) but different trees (i.e. different ways of joining branches), and in the absence of experiments to determine if any one linkage method better reconstructs the real structure in a dataset than any other, the primary criterion for choosing a linkage method should be the intelligibility of the tree that it produces. Is it easy to identify clusters? Is the relationship among clusters easy to discern?

The differences among the three linkage methods can be demonstrated by clustering the same dataset with each of the three methods in Acuity (Figure 13a).

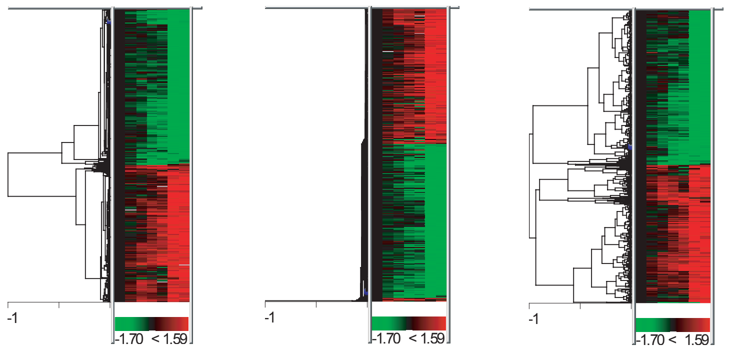


Figure 13a (i), (ii), (iii)

Average Linkage (i)
The distance between clusters is determined by the distance between 'average neighbors'.
Single Linkage (ii)
The distance between clusters is determined by the distance between the 'nearest neighbors'. This linkage method tends to produce dendrograms containing long chains.
Complete Linkage (iii)
The distance between clusters is determined by the distance between the 'furthest neighbors'.

From this example, which is in no way contrived or atypical, one notices two things:

- On the gross scale, the color tables are very similar.
- It is relatively straightforward to rank the methods in order of decreasing intelligibility: Complete, Average, Single.

The differences between the methods are even more striking if we look at three trees generated on a different dataset. In the following case, we have a dataset consisting of 31 substances and 39 microarrays, where the microarrays fall into two distinct classes. Both substances and arrays were clustered, but for simplicity we show only the array tree (Figure 13b).

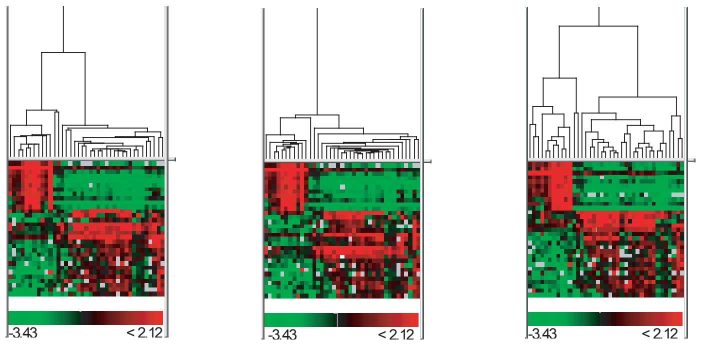


Figure 13b (i) Average, (ii) Single, (iii) Complete

As in the first example, the color maps are basically identical. However, there is again an obvious difference in intelligibility among the three methods, which is the same as we came to in the first example: Complete, Average, Single. In the dendrogram constructed by Complete linkage, the clusters have an appropriate amount of nesting, and an appropriate amount of separation. When using Average and especially Single linkage, the clusters are so tightly nested that it is difficult to make any sense of their internal structure and their relationships to each other.

Having said all that, the data may be better represented by a single-linkage tree than by a complete linkage tree. However, in science one should always begin with the simplest explanation, and complete linkage provides by far the simplest structure of the three methods.

5.4.3 Branch Ordering

Before we can go further, we need to define some terms used to describe the dendrogram (which is a mixture of the arboreal and the genealogical):

- The objects that are clustered are **leaves**.
- A cluster containing at least two objects but not all of them is a **branch**.
- A branch-point on a tree is a **node**. The nodes beneath a node are its children; the nodes above a node are its parents.

The position of a node in the tree represents the similarity between its two child nodes. The output of the algorithm, however, leaves one part of the representation unspecified: which way to order the child nodes beneath a node. For example, consider the following four dendrograms:

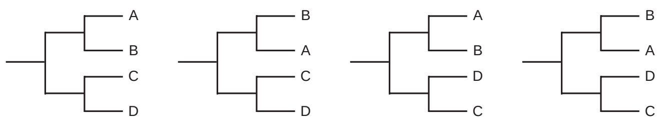


Figure 14

Looking at the first dendrogram, we can say that the node (A,B) is similar to the node (C,D). However, between the nodes, we do not know whether A is more similar to C, or to D, and the algorithm does not tell us this. Therefore, each of the four branches above is an equivalent representation to the clustering solution provided by the algorithm.

In a large dataset, this underdetermination of branch ordering can have a substantial effect on the appearance of a tree, and hence of the intelligibility of the solution. For this reason, Acuity includes a number of methods of automatically swapping branches so that they match as closely as possible some other ordering, such as the ordering provided by a SOM or a PCA.

5.5 Non-hierarchical clustering

As we saw in Dataset Filtering and Management, the non-hierarchical clustering visualization in Acuity is an extremely effective method of summarizing the main expression signals in a dataset. For completeness, let's describe what we see in such a visualization:

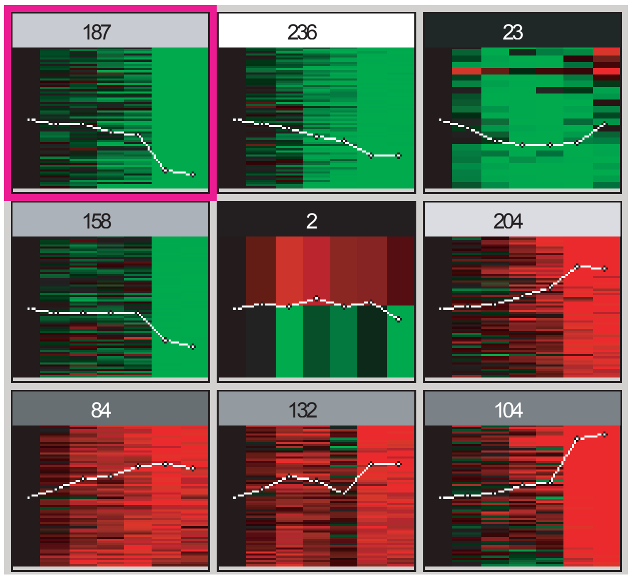


Figure 15

This is a visualization of a Self-Organizing Map analysis, consisting of nine clusters. The title bar of each cluster reports the number of substances in each cluster, and each title bar is colored on a grayscale from black to white, depending on the relative number of substances in each cluster.

The cluster with the pink square around it has been selected, which means that the 187 substances in it are selected everywhere else in the Acuity interface.

The trace drawn on each cluster shows the average profile of all the substances in the cluster.

The colors in each cluster are formed from the values of each substance on each microarray.

Because this is a Self-Organizing Map visualization, the clusters are arranged by similarity on the 3×3 grid. That is, similar clusters are together, and clusters diagonally opposite have opposite average profiles.

But while the visualization is the same for all four algorithms—K-Means, K-Medians, Self-Organizing Maps, Gene Shaving—the outputs of each of the algorithms are different. In order to interpret the visualizations, therefore, we need to know a little more about each of the algorithms.

Each of these clustering methods shares the same pre-processing options, including the choice of using an automatically generated or a fixed random seed. The default setting uses an automatically generated random seed. With this option, if you run the same analysis twice in a row, you will obtain a slightly different result, usually manifested by the cluster memberships being slightly different. That is, you will see roughly the same clusters created, but there will be slightly different numbers of substances in each cluster.

This variation in cluster membership across repeated analyses can be a little disturbing, especially if you are still harboring under Preconception 3. You may think that this is yet another layer of arbitrariness that is imposed during the analysis that you do not need. But consider the alternative: would you be any more confident in a cluster result if you always used the same seed, instead of an automatically generated seed? It is still a random seed. Furthermore, you still have the arbitrariness of choosing one among many distance metrics.

We think it is much healthier to embrace the small variations in cluster membership and look at clustering in a slightly more sophisticated manner. It is not unusual to find substances that belong only marginally to a cluster, and that change membership as the initial conditions are varied among repetitions of the analysis. This can mean a number of things:

- You have chosen the wrong number of clusters to find in the dataset, so that similar substances are forced into different clusters, and then move among these similar clusters among solutions.
- There may be no clear separation of substances into clusters in your dataset (this is unusual in a gene expression experiment).

In the Self-Organizing Map visualization above, you can see from their profiles that the first two clusters are very similar, as are the first two on the bottom row. It would be entirely expected that if we ran this analysis a second time, the overall solution would be similar (i.e. the profiles found would be similar) but that cluster membership would change a little among the two sets of similar clusters. This is clearly a case of partitioning the dataset into more clusters than are supported by the data.

5.5.1 Gap Statistic

In non-hierarchical clustering we tell the algorithm how many clusters to find in a dataset. You may protest, and with good reason: what is the point of having a clustering algorithm if I have to tell it how many clusters to find?

While we have focused quite strongly on the arbitrariness in clustering in order to counter the various preconceptions about it, it is possible to go too far in the other direction. That is, we should not deny that many datasets have a large amount of intrinsic structure, and that clustering should be able to find it.

Consider the dendrogram in Figure 16. At the grossest level, we can see that the dataset can be divided into two main clusters: one at the top that is predominantly green, and one at the bottom that is predominantly red. This division is reflected in the tree, which divides into two main branches. At any particular correlation value you can

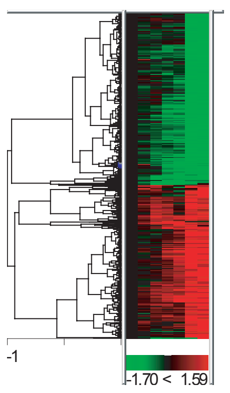


Figure 16

draw a vertical line through the tree, and the number of branches that it crosses is the number of clusters that the dataset is partitioned into at that value. The dendrogram does not tell us how many clusters are in the data: we just have various numbers of clusters with various degrees of similarity.

The problem we are facing here is one of granularity: at one level, there are obviously two clusters, but if we look more closely we can divide those two clusters further. we move down the tree, we see that three clusters, five clusters and nine clusters are relatively stable partitions (Figure 17).

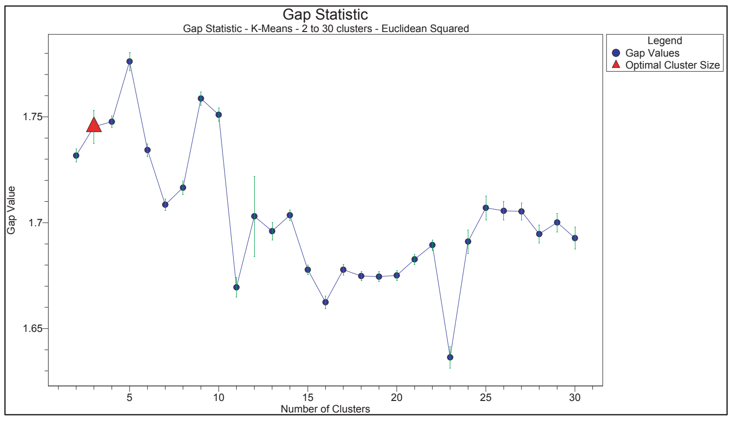


Figure 17

Which is the best solution? There are objective ways of answering this question, where one measures the within-cluster dispersions, such as the Gap Statistic. To some extent, the quality of a solution depends on how well it matches your interests. If you're only interested in the gross structure, then two clusters is a good answer. If you're interested in the fine detail, you might want to see more clusters.

Going back to our SOM example above, this seems to be a case where 9 clusters is not the best partition: on the gross scale, two clusters is better, while if you're interested in a finer partition, five appears to be a stable partition.

The Gap Statistic is one method of estimating the optimal number of clusters in a dataset. It consists of two parts:

- a metric for estimating the goodness of a partition of a certain size (the Gap value);
- a rule for determining the best Gap value.

When you perform a Gap Statistic calculation, you specify the range of clusters to test; for example, you compute a Gap value for every cluster solution between 2 clusters and 20 clusters. The Gap value chosen to be the best is not necessarily the one with the largest Gap value; it is the first local maximum. For more details, see the printed manual.

5.5.2 K-Means and K-Medians

Having described the general strategies of non-hierarchical clustering above, there is very little more to say about K-Means and K-Medians clustering except that they are the fastest non-hierarchical methods in Acuity. The algorithms are similar to the Self-Organizing Maps algorithm, without the clusters being constrained on a 2-dimensional lattice. Consequently, the clusters are arranged in random order on the visualization. When creating solutions with large numbers of clusters, K-Means and K-Medians can be difficult to interpret. On the other hand, they are very effective for analyzing large datasets into small numbers of clusters.

5.5.3 Self-Organizing Maps

Like K-Means and K-Medians, the Self-Organizing Maps algorithm uses a similarity metric to group substances into clusters. Unlike the K-cluster methods, the Self-Organizing Maps algorithm arranges clusters on a two-dimensional grid, generated by two unobserved latent variables. This makes the visualization much more informative than in the case of the K-cluster methods, especially when one finds a large number of clusters.

5.5.4 Gene Shaving

Unlike K-Means, K-Medians and Self-Organizing Maps, the Gene Shaving algorithm uses Principal Components Analysis and the Gap Statistic to generate its clusters. Because of this and other fundamental differences in clustering methodology, it has the following differences as well:

- It cannot find more clusters than there are microarrays in the dataset.
- Its clusters are not mutually exclusive: a substance can be in more than one cluster.
- Substances with opposite signs are clustered together. That is, a substance showing linear increase along the arrays is likely to be clustered with a substance showing linear decrease along the arrays.

For these reasons, it is often worth doing a Gene Shaving analysis of a dataset in addition to one of the other non-hierarchical methods. It can produce some very interesting results.

5.6 Principal components analysis

Of all the advanced analysis methods in Acuity, Principal Components Analysis (PCA) causes the most confusion. While the confusion may arise from a number of different sources, it is often caused by the following:

- **Cluster confusion**. Strictly, PCA is not a clustering method. It is a data reduction method.
- **Visualization vagueness**. Novices find the PCA visualization difficult to interpret.

Let us begin by dispelling these sources of confusion.

5.6.1 PCA is not clustering

Suppose we have a dataset consisting of 1000 substances and their values on 10 microarrays.

If we do a cluster analysis of this dataset, we divide the substances into groups (clusters), but we still have 10,000 data points.

If we do a PCA and specify, say, to find 3 principal components, we end up with 1,000 substances and their scores on 3 principal components. We have 3,000 data points, which is less information than when we started. That is why PCA is known as a data reduction method.

After a PCA, we could save the PCA analysis result as a new, reduced dataset and then go on to cluster that. In some types of data mining where datasets are very large, that is indeed what happens. At present most microarray datasets are not that big, and reducing a dataset by PCA to then cluster it has been shown to be no more effective than clustering the raw data.

What we do instead is plot the substances on the components and use that visualization to explore the data. At this stage, we do treat PCA like clustering, as we look for clusters in the data as represented in the PCA space.

5.6.2 Visualization vagueness

The PCA visualization is very different to the cluster visualizations, a fact that can compound confusion about PCA. However, it is really not very complicated.

Consider the example we described above: the result of the PCA analysis is a table of data with dimensions 1000 substances × 3 component scores. Thus every substance can be plotted in the 3D space of the Acuity PCA visualization.

If you perform a different PCA analysis and find, for example, 5 principal components, then the result is a table of 1000 substances × 5 component scores, and the Acuity visualization gives you the option of choosing which 3 components to plot.

5.6.3 So what does it all mean?

If you open a PCA analysis result you might get something that looks like Figure 18.

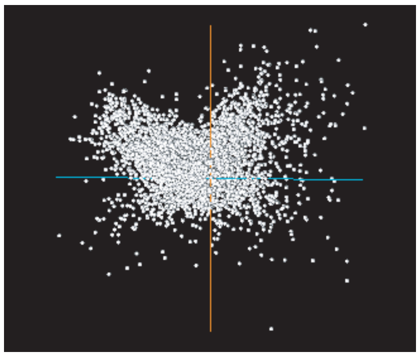
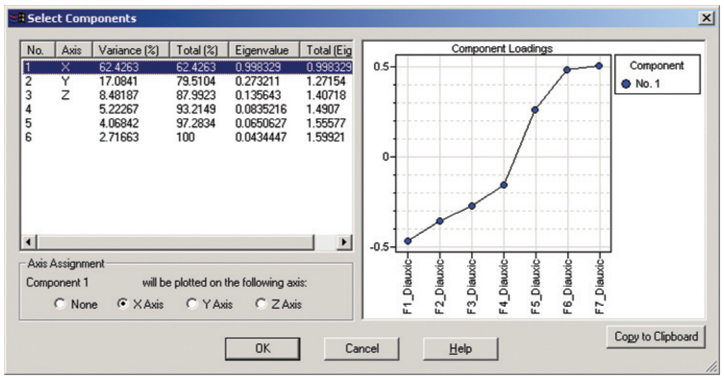


Figure 18

If you select Properties from the right-mouse menu, you will see something like Figure 19.



No.	Axis	Variance (%)	Total (%)	Eigenvalue	Total (Eig
1	X	62.4263	62.4263	0.998329	0.998329
2	Y	17.0841	79.5104	0.273211	1.27154
3	Z	8.48187	87.9923	0.135643	1.40718
4		5.22267	93.2149	0.0835216	1.4907
5		4.06842	97.2834	0.0650627	1.55577
6		2.71663	100	0.0434447	1.59921

Axis Assignment

Component 1 will be plotted on the following axis:

( ) None [✓ X Axis] ( ) Y Axis ( ) Z Axis

Figure 19

This dialog box reports the details of the PCA result, but is also used to display component loadings (more about this below) and to configure the main 3D display.

One of the aspects of PCA that we have not yet discussed is how the components are chosen, and how they are related to each other. There are two things you should know about the components:

- As you can see from the table in Figure 19, the components are chosen based on how much variance they can explain in the sample; as you go down the list of components, they explain less and less variance.
- Each component is independent of the previous component.

What do these two points mean in practice? This analysis result has six components. The first component explains over 62% of the variance in the sample. If you look at the component loadings graph on the right of Figure 19, you can say that substances matching this profile contribute to 62% of the variance in the sample. Because this is the first component, and so substances with high values on this component are plotted at the far right of the X-axis on the main PCA display, if you select those substances on the PCA display and then look at their profiles on the Graph tab, you will see something like Figure 20.

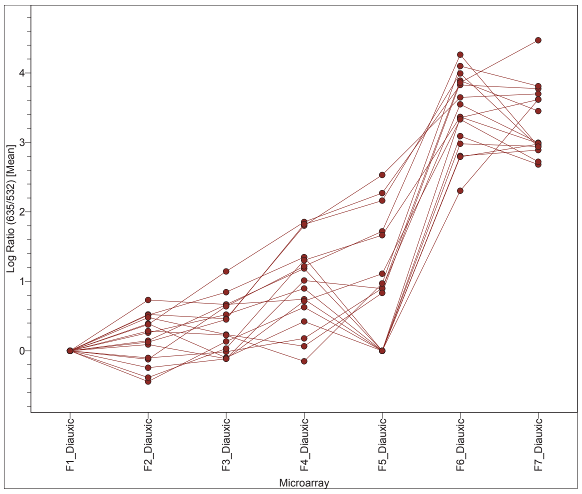


Figure 20

You can see that these profiles are indeed similar to the component loadings graph of principal component 1, and that they do show large changes over the time course in this experiment.

You can think of the component loadings graph for each component as the profile of a 'supergene' that is controlling the experiment. In this experiment, for example, with only 6 principal components we can explain 100% of the variance in the sample. That means that we can reduce the complexity of this experiment to 6 basic expression profiles, which look like the graphs of the loadings of the 6 principal components.

The profile of each real substance in the experiment can be thought of as some combination of the profiles of the supergenes. If a substance has a profile very much like the loading graph of component 1, it will be plotted close to the X-axis on the main PCA display, with only small contributions on the Y-axis or Z-axis.

Describing PCA in this way, it is actually a little bit like clustering, only it is grouping substances according to how closely they match certain ideal profiles that explain the most variance in the sample.

5.6.4 Plotting microarrays

For one important class of microarray experiments, such as those represented by cancer studies, we are interested in similarity among microarrays as well as similarity among substances. In the screenshot of the PCA visualization in Figure 18, we have plotted substance component scores. We could equally plot microarrays, although in a time course experiment that tends not to be very informative.

In studies where we are attempting to find similarities among microarrays, plotting microarrays is very informative.

The PCA visualization in Figure 21 is from an experiment where each microarray is from a different cancer cell line. Cell lines of specific types are colored similarly (based on the colors of dataset quicklists): red, leukaemia; green, colon; pink, breast; light blue, lung; orange, ovarian; yellow, renal; grey, CNS; brown, melanoma.

The first thing you notice is that some cell lines cluster together very strongly, such as leukemia and colon, while others are spread out. This means that some cell lines must be distinguished by the expression of a certain small number of substances with similar profiles, while others are characterized by substances having a variety of expression profiles.

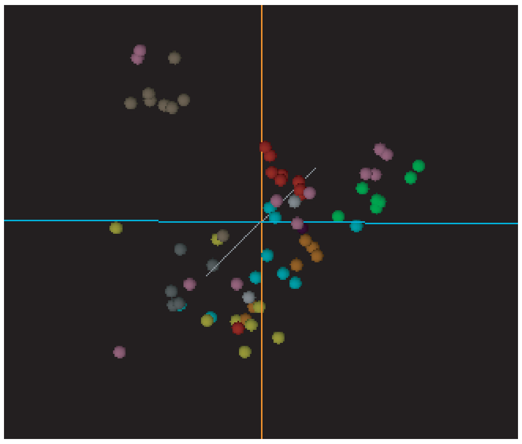
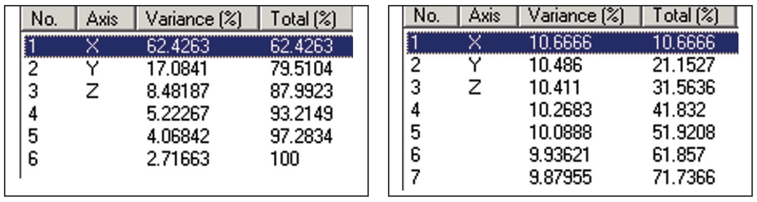


Figure 21

Of the cell lines in this experiment, only the breast cancers occur in all four quadrants of the PCA, meaning that substances with diverse profiles are contributing to them. Even those samples not as tightly grouped as leukemia and colon are nevertheless confined to one or two quadrants: renal, CNS and melanoma are all at the top, for example. On a dendrogram from this dataset (not shown here) these three groups of cell lines all fall under one main branch, while colon, leukemia and breast are on another.

5.6.5 Using PCA as a precursor to clustering

At the beginning of this section we formulated Preconception 3 ("clustering will find the real structure in my dataset") and showed how clustering finds structure even in random data. PCA is not as easily fooled by random data as clustering. Figure 22 shows the variance calculations from a PCA of biological data (left) and random data (right).



No.	Axis	Variance (%)	Total (%)
1	X	62.4263	62.4263
2	Y	17.0841	79.5104
3	Z	8.48187	87.9923
4		5.22267	93.2149
5		4.06842	97.2834
6		2.71663	100

No.	Axis	Variance (%)	Total (%)
1	X	10.6666	10.6666
2	Y	10.486	21.1527
3	Z	10.411	31.5636
4		10.2683	41.832
5		10.0888	51.9208
6		9.93621	61.857
7		9.87955	71.7366

Figure 22

There are two main differences between the biological data and random data. The first is that the variance explained by the first component of the biological data is very much greater than its counterpart from the random dataset. The second is that in the random dataset all the components explain roughly the same amount of variance.

In a random dataset, one expects that each component explains the same percentage of variance, and that this is equal to 100/(number of microarrays). In this case where we have a random dataset consisting of 10 microarrays, we would expect that each component would explain around 10% of the variance, and this is exactly what we see. In the biological dataset of 7 arrays, we see that the first component explains a great deal more variance than one would expect from random data (whether it is relevant biological structure or not is another question). Furthermore, the second component explains a very large proportion of the remaining variance in the dataset (38%) so it is also significant.

The deviation from random is very easily seen in the PCA visualization: random data produces a symmetric, randomly-distributed cloud of points, while a biological dataset tends to be asymmetrical. (Note that if you center and scale rows before the PCA, the result will have some spherical symmetry, although you will still be able to discern structure in biological data).

It is very unusual for a biological dataset not to show significantly more variation than random. However, principal components analysis is an essential tool that you can use to reassure yourself that the variation you are seeing is significant.

6 Statistical Analysis

In the Dataset Filtering and Management topic we saw some simple methods of filtering datasets based on, for example, fold-change conditions. However, as pointed out in that section, a fold-change filter is a blunt instrument: it can remove substances from your dataset a little too indiscriminately. There are more sophisticated and statistically rigorous ways of finding the differentially expressed genes in a dataset than simply looking at those with the biggest fold changes.

6.1 Significance Statistics

Acuity offers some statistical methods for identifying the genes that change most significantly in an experiment. While they are relatively straightforward, they are also very effective when applied to microarray data. However, in order to use them intelligently, one must understand exactly what they are measuring, and what one can infer from them.

6.1.1 Two-Sample t-Test

Student's t-Test as implemented in Acuity compares the means of genes in two groups of microarrays. It is testing the null hypothesis:

- The substance is not differentially expressed between two groups of arrays, or
- There is no difference in the substance's mean expression between two groups of arrays.

The p-value is the probability that the null hypothesis is true. A low p-value, therefore, suggests that the substance is differentially expressed.

Student's t-Test compares the variation between the groups (i.e. the differences in their means) with the variation within the groups. This means that substances will have lower p-values if they satisfy two conditions:

- They have different means between the two groups;
- They have similar values within each group.

On the demo Acuity dataset, for example, when we look at the microarray parameters tab we notice that the glucose level in the experiment decreases dramatically between time points 5 and 6. Use the t-test to find the substances that are differentially expressed between the first five time points, and the last two time points. If we plot their profiles we get Figure 23.

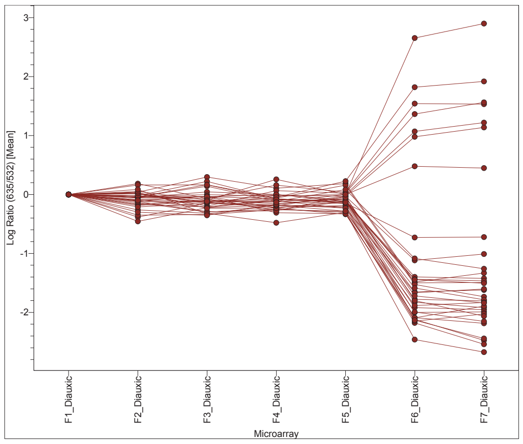


Figure 23

The 30 substances that score most highly (i.e. that have the lowest p-values) satisfy the two conditions mentioned above: their means are very different between the two groups, and within each group their values are very similar.

If we look at the substances that have the lowest p-values between the first four microarrays and the last three, we get a very different group:

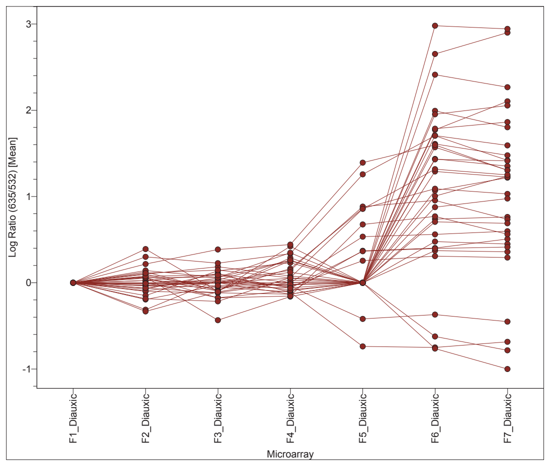


Figure 24

Note, however, that the substances satisfy the same general requirements: within-group uniformity, between-group difference.

Note also that the substances in this second case are expressed at quite low levels: most of them show less than two-fold change. One of the benefits of t-tests over fold-change tests, for example, is that they identify substances with small but significant changes in expression.

In a time course experiment, therefore, the t-test is useful for finding substances that are differentially expressed between two different time points, where each time point is represented by a number of microarrays (replicates). Where you do not have replicates, as in the examples above, you can find substances that are differentially expressed between sets of time points. T-tests are not useful for answering the general question, "In a time course experiment, which substances are differentially expressed?" as this question does not specify two groups of microarrays.

In an experiment where the microarrays are from two different experimental conditions, however, the t-test can be extremely useful for identifying substances that are differentially expressed between the conditions. The two conditions naturally define two groups of microarrays, and the t-test will identify substances that have similar expression levels within each group, and different mean levels between the groups.

6.1.2 Analysis of Variance

Analysis of Variance (ANOVA) is a generalization of the two-sample t-Test to many groups. It is not unusual to have more than two treatments or two types of samples in a microarray experiment, and in such cases one would use an ANOVA instead of a t-Test.

6.2 Using Principal Components Analysis

As explained in the Clustering topic, Principal Components Analysis (PCA) transforms a dataset so that substances are scored against ideal profiles that explain the most variance in the dataset. Substances that score highly on the first few principal components, therefore, have the following properties:

- They show high correlation with profiles that explain the most variance;
- They show a large change across the dataset.

PCA, therefore, is an excellent way of identifying substances that are changing the most in a dataset.

You can see the "ideal profiles" graphed in the PCA Properties dialog box (which you can open by double-clicking on a PCA display). For each component that you select, the component loadings are drawn (Figure 25).

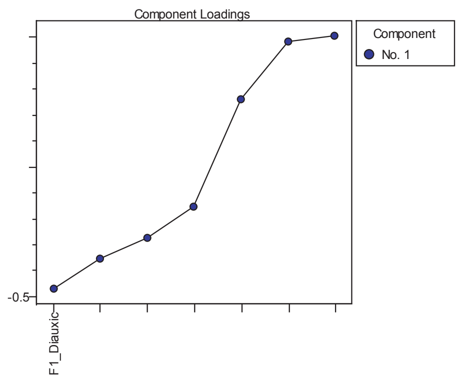


Figure 25

If you then go to the PCA display and select the 30 substances that score highest on the X-axis, you will get Figure 26.

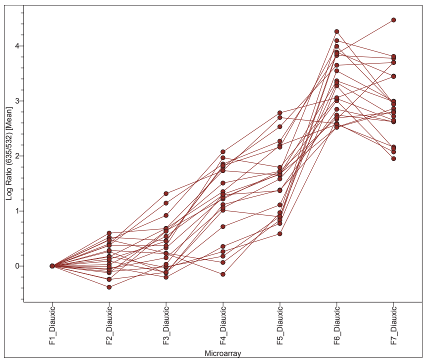


Figure 26

As you can see, these substances have profiles that correlate closely with the component loadings graph of principal component 1. They also have very large fold-changes: up to 4-fold.

If we choose the substances that score highest on the X-axis in the negative direction, we get Figure 27.

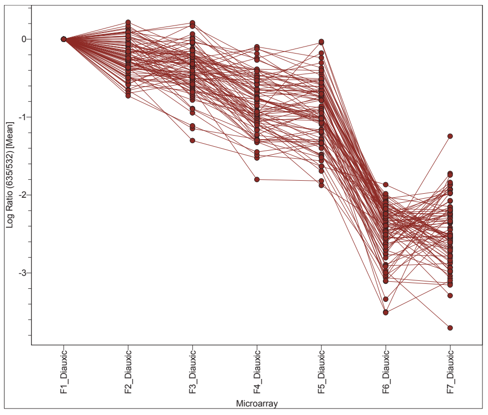


Figure 27

They are also changing up to 4-fold, but in the other direction.

Clustering methods will identify these groups of substances. For example, if we do a 2×2 self-organizing map of the dataset, we find Figure 28.

The clusters at the top right and the bottom left correspond to the substances that score highly on the first principal component (and the other two correspond to the second principal component). If we did a 5×5 SOM, we would see clusters containing substances corresponding to further principal components.

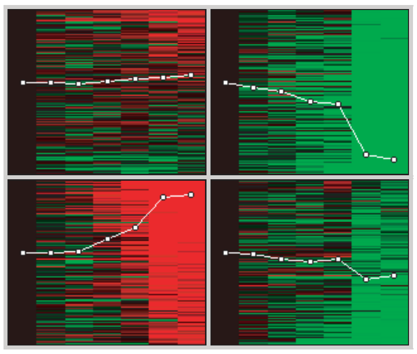


Figure 28

What PCA gives us that a clustering method does not is a measure of significance. The PCA:

- Ranks the components by the amount of variance that they explain, and
- Scores genes according to how well they correlate with the component.

In a self-organizing map, the substances that change most tend to be grouped in clusters around the outside of the SOM, with the largest changing clusters in the corners:

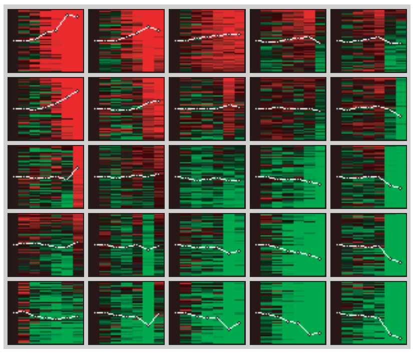


Figure 29

However, that still doesn't tell us which of the corner clusters is most significant, or how to rank the substances within each cluster. PCA gives us both.

6.3 Match Expression

So far in this section we have been looking at techniques for mining datasets for significant substances without using any prior knowledge of what we might be looking for. However, sometimes we know what we're looking for: we have applied a known treatment to our samples, or we are familiar with some of the substances that are going to be differentially expressed and we want to find others that might be co-expressed. In such a case, Match Expression can be a useful tool.

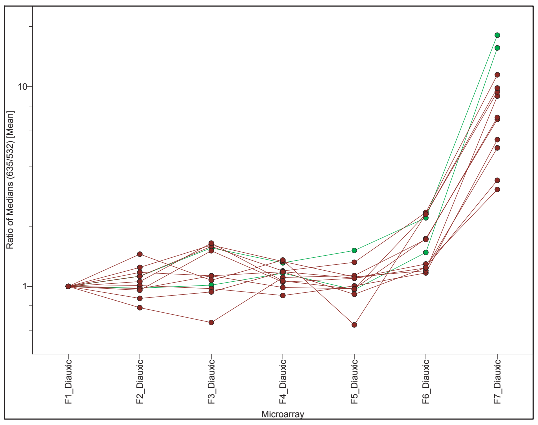


Figure 30

Match Expression orders the main data table in Acuity according to how well each gene correlates with a selected substance (or substances).

For example, suppose we have two substances that we know are co-expressed, and we want to find others that have the same profiles as these two. Match Expression is designed precisely to answer this question. After it orders the substances in the table by correlation with our two co-expressed genes, we are free to choose our own correlation cutoff, and select the substances that pass the cutoff, as shown by their profiles below (where our two initial co-expressed substances are in green):

If we want to see how important these genes are, we can look at where they are plotted on a PCA display of the dataset:

We see that most of them score highly on components one and two, indicating that they are explaining a large amount of variance in the experiment.

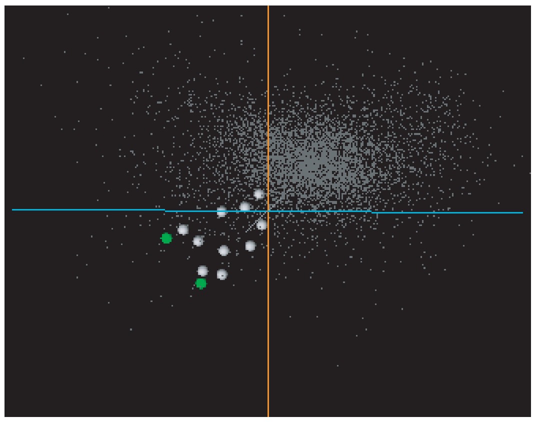


Figure 31

We can also see substances that are anticorrelated with our co-expressed substances: as expected, these are also significant in the PCA, and fall upon the opposite vector to the earlier group of substances.

7 Conclusion

A number of the analysis examples used in this application note (Figures 13, 15-20, 22-31) were generated from the data from the first full-scale study to use microarrays, DeRisi, J., Iyer, V., and Brown, P.O. "Exploring the Metabolic and Genetic Control of Gene Expression on a Genomic Scale", Science 278 (1997) 680. In that article the authors investigate the diauxic shift in yeast (the change from anaerobic to aerobic metabolism) with a view to understanding the genomic basis of this metabolic change.

DeRisi et al. did not have the sophisticated software that we now have for microarray analysis, and yet the significant expression changes and profiles that they report in their Figure 5 are exactly those that we see in the various self-organizing maps clusters of Figure 29: as time increases along the X-axis, we see groups of genes with increasing expression, and groups with decreasing expression. Furthermore, the patterns that they identify as showing the greatest fold changes and hence the greatest variance are precisely those that are picked out by the principal components analysis, as shown in our Figures 26, 27 and 30.

The first lesson to take from the comparison of manual methods with algorithmic techniques is that many of the clustering and other analysis tools that we now have at our disposal do nothing more than what a biologist with a keen eye can do when looking at a dataset. The software identifies patterns with very little effort and displays them in a way that makes them easily interpretable, but it is nevertheless doing what you could do by eye, given enough time and patience. The benefits of using powerful software are not to be underestimated, for we can now do in an hour what previously may have taken a month, and on very large or complex datasets manual methods become practically impossible. However, it is worth remembering that much of what the software does is relatively straightforward. By explaining and demystifying analysis algorithms, we hope to return control of data analysis to the biologist.

The centerpiece of DeRisi et al is not Figure 5, with its identification of expression profiles and fold changes, or even the elaborate Figure 3, which shows the metabolic changes during the diauxic shift in the context of a number of interconnected metabolic pathways. Rather, it is the intimate understanding of yeast biology that is evident in every paragraph discussing the significance of the microarray data.

The second lesson to take, then, is that no matter how much data you gather from a microarray experiment, no matter how many clusters or other analyses that you generate with your software, the analyses must be incorporated into an understanding of the biology of the system that you are studying. Microarrays are very powerful tools, but they are only tools used to the end of understanding the biology. Microarray analysis software does not and cannot replace the hard work of integrating your microarray data with the diverse knowledge of your biological system, and producing a coherent functional story.

In this application note we explained the use of some statistical tests by applying them to DeRisi's diauxic dataset. Figures 23 and 24, for example, show the profiles of genes that according to a t-Test have highly significant changes between the first five time points and the last two time points in the diauxic dataset (the two groups corresponding to the two different metabolic states). In the years since the DeRisi et al. study was published we have learned a great deal about microarray data analysis, and in particular we have learned that previously ignored genes with small fold changes are not necessarily biologically insignificant.

However, the third lesson to take away is the contrary of this, namely that statistical significance does not imply biological significance. Just as you are not guaranteed of finding all the biologically significant genes by using a fold-change filter, neither can you be certain that all the statistically significant genes are biologically significant. Statistical significance is a rather simple measure; it tells you something about the consistency and reproducibility of your results, but it says nothing about whether those results are biologically meaningful. Hidden systematic errors, for example, can still make a mockery of significance statistics.

No matter how you decide on a list of interesting genes, your job does not end there. Data analysis software like Acuity provides a toolbox of statistical and data mining tools, none of which claims to be exhaustive. Each tool gives you access to a different part of the dataset. Use all the tools, and you will be better able to construct a complete picture of your biological system.