Babraham Bioinformatics

Advanced Analysis with SeqMonk

Version 1.3.0

Licence

Introduction

Once you have been working with SeqMonk for a while you will quickly master the basic controls and plots. The real trick to making effective use of the program though is how you apply the tools you have to your data.

This course is intended as a follow on to the basic SeqMonk course and goes into some of the more advanced features of the program, as well as providing some advice on how the tools can best be applied to your data. This course focusses more on handling biases within your data and how to apply more rigorous statistical analyses – including those applicable to more complex experimental designs such as time courses or dose responses.

As with the original course I've tried to keep the advice here as neutral as possible with regard to any specific type of experiment. The topics presented here should be of use for the analysis of most kinds of data since they address generic problems from sequencing experiments.

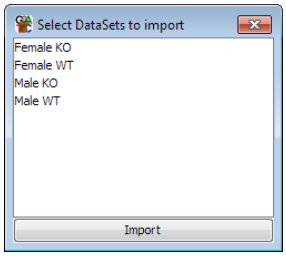
Data Import

Whilst basic data import is hopefully pretty simple, there are a few additional options which you may not have tried and which might make life easier.

Reimporting Data

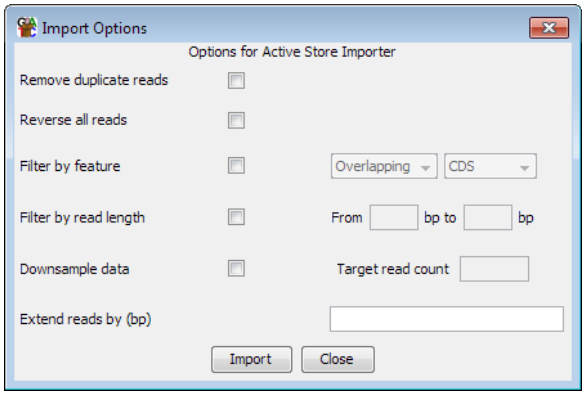
Rather than importing data from a mapped file you can reimport it from the current SeqMonk project or a different project.

Cross Importing Data



To cross-import data from a different SeqMonk project you can select File > Import Data > SeqMonk Project. This will then check to see that the genome assemblies match between the other project and your current project and will then find the list of DataSets in the project from which you can select those you wish to import. You can copy over as many or as few as you like into your current project.

Reimport within the same project



You can also choose to reimport a dataset you already have in your current project. This will duplicate the data in the project, but also allows you to select some of the other import options at the same time. Modifications you can make when re-importing would include deduplication, extension, or reversing the strand of all reads. You can also filter the re-imported data either by keeping or rejecting reads which overlap with a class of features, or by more generally down-sampling the data to keep a random subset of what you started with.

HiC Import

One of the newest features in SeqMonk is the ability to handle HiC association data. HiC data comes as pairs of reads, where there is no expectation that the reads sit physically close together, or are even on the same chromosome. Analysis is based on the degree of association of parts of the genome based on the number of pairs of reads whose ends come from the regions being compared.

HiC import is now a standard option from any of the import tools. There is no special format for HiC data, but the import filter assumes that all of the data comes in pairs of HiC associated positions. This means that to create a mapped HiC file you should only include sequences where both ends were mapped and the mapped positions should be placed immediately after one another in whichever file format you choose to use.

There is very little SeqMonk can do to check that your HiC data was correctly formatted, so if you have mismatched your sequences you are unlikely to see an error from the program. The best way to check whether the data is OK is to run a HiC heat map following import to see what you get. Heat maps should produce a characteristic pattern, and if your data seems to be randomly distributed then you should be suspicious that the data import failed.

During HiC import there are some options you can apply to filter the data as it comes in. For the most part filtering is better done at an earlier stage during mapping using programs such as HiCUP, however some filtering can be applied at import. Specifically you can choose to remove short interaction pairs, or you can choose to ignore trans hits (pairs which sit on different chromosomes) to significantly reduce the size of your dataset if you're specifically looking at cis interactions.

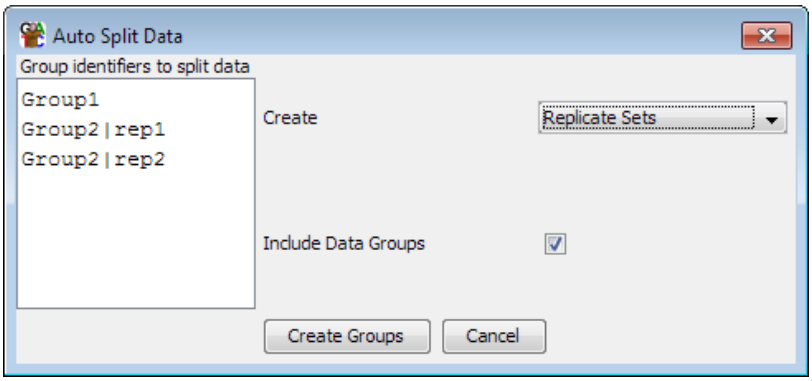
Automatic sample grouping

After importing data into SeqMonk you should create appropriate sample groups and replicate sets to group your samples together. The two are used for different purposes:

- Data Groups merge the raw data for two or more data sets and make them appear as if they had been imported from a single file.

- Replicate Sets average the quantitation for two or more data sets or groups, but retain the distribution of quantitated values for use in statistical tests. Replicate sets should be used to mark biological replicates of each experimental condition.

Although you can make data groups and replicate sets manually using the options under the data menu, there is a quicker way to create them if your data set names have a consistent naming scheme using the option Data > Auto Create Groups/Sets.



In the box on the left you can put in a set of text strings which can be found in the names of your data sets and which mark the groups you want to create. The tool will then look for a match to each string and will make either a data group or a replicate set out of the samples which match. If you need to use more than one discontinuous string match to define your groups then you can separate strings with bar characters to make more complex patterns. Groups made this way can always be edited with the traditional data group and replicate set editors.

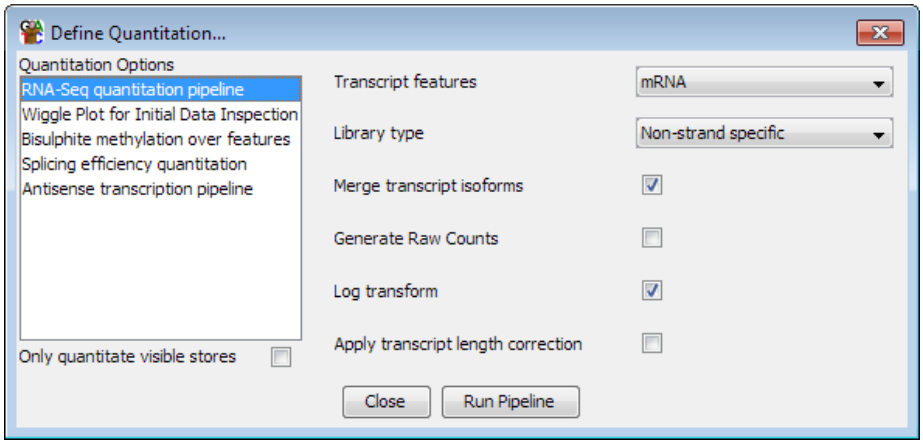
Quantitation Pipelines

The traditional way to quantitate data in SeqMonk is to use a two step process where the first step involves the creation of a set of probes over regions which will later be quantitated, and the second step assigns a value to each probe for each dataset based on the data.

In some cases there are types of quantitation which can't be performed in this type of 2-step process, and there are also some common quantitations for which it would be convenient to have a simpler way to perform them. These use cases are what Quantitation Pipelines were designed for. They are automated quantitation processes which can perform probe generation, quantitation and potentially even filtering in a single step. Quantitation pipelines can be accessed under Data > Quantitation Pipelines.

RNA-Seq quantitation pipeline

The RNA-Seq pipeline uses a type of quantitation which can only be perfomed within a pipeline since it requires that quantitation and probe generation are performed simultaneously. The basic premise of the pipeline is that it is a read count quantitation over a set of features, but for multi-exon features it only counts the reads which sit over the exons of the feature and ignores those in the introns.



The options you have within the pipeline allow you to choose which class of features you're going to use to quantitate your data (see the later notes about filtering features based on their biotypes). You can also choose the type of strand specificity in the libraries you're quantitating so you can only count the relevant reads.

The default quantitation from this pipeline is log2 RPM (reads per feature per million reads of library), but you can alter this. If you're going to use the quantitation in an external analysis tool such as DESeq or EdgeR then you will need uncorrected raw read counts, so there is an option to generate these. If you want to compare the expression levels of different genes within the same sample then you can also correct by read length to get log2 RPKM values, but this is not recommended when comparing expression values between samples.

By default the RNA-Seq pipeline merges together the exons of different transcript isoforms for the same gene to give you a single per-gene expression value. You can choose to get output for each transcript isoform by unticking the 'merge transcript isoforms' box, but SeqMonk does not try to do a likelihood based assignment of reads to a single isoform, so reads which map against more than one isoform will be counted more than once in this mode.

Wiggle plot pipeline

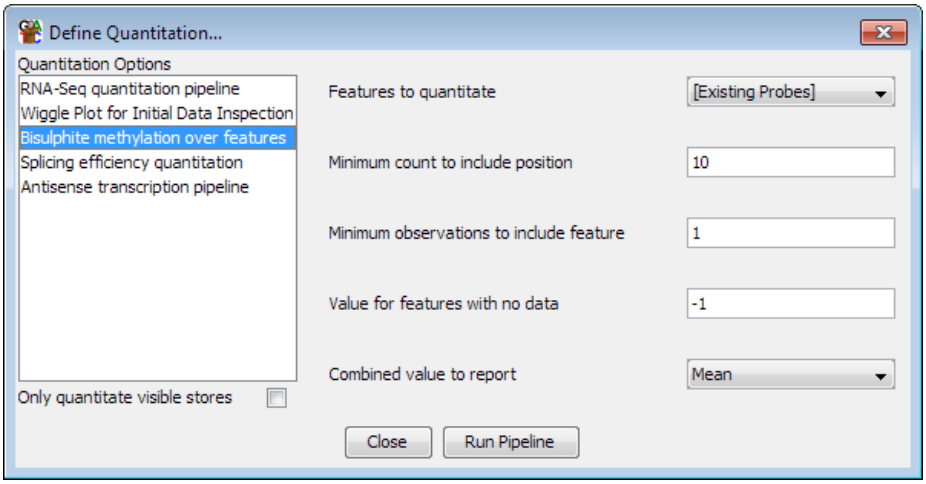
The wiggle plot pipeline is a convenience method, and everything in this pipeline could be performed in the normal quantitation options. This simply generates a set of running window probes and quantitates them with a corrected read count. This is an easy way to generate an initial unbiased quantitation of your data to look quantitatively at your read counts over a region of interest.

You can choose which region you want to analyse (just what you're looking at now, the current chromosome or the whole genome) and the pipline will try to select a suitable window size for your probes.

Bisulphite methylation over feature

When analysing bisulphite data what is imported into SeqMonk are not the read positions, but the individual methylation calls. In a BS-Seq dataset each 'read' is only 1 base long and the strand of the read indicates the methylation state (forward = methylated, reverse = unmethylated).

You can calculate a percentage methylation value for a region of the genome by simply counting the total number of methylated and unmethylated calls within that region, however this has a number of problems. You could have very few calls which could result in a very unreliable overall methylation value. You could also have very biased coverage such that the majority of your reads come from one or two positions within the region which again might not produce a good overall average.



The bisulphite methylation pipeline provides a way to account for some of these problems. It provides a way to filter each call position by the degree of coverage and then produces an overall methylation value which weights each valid call position equally. It also uses a flag value to mark those positions for which an overall level of methylation could not be determined reliably.

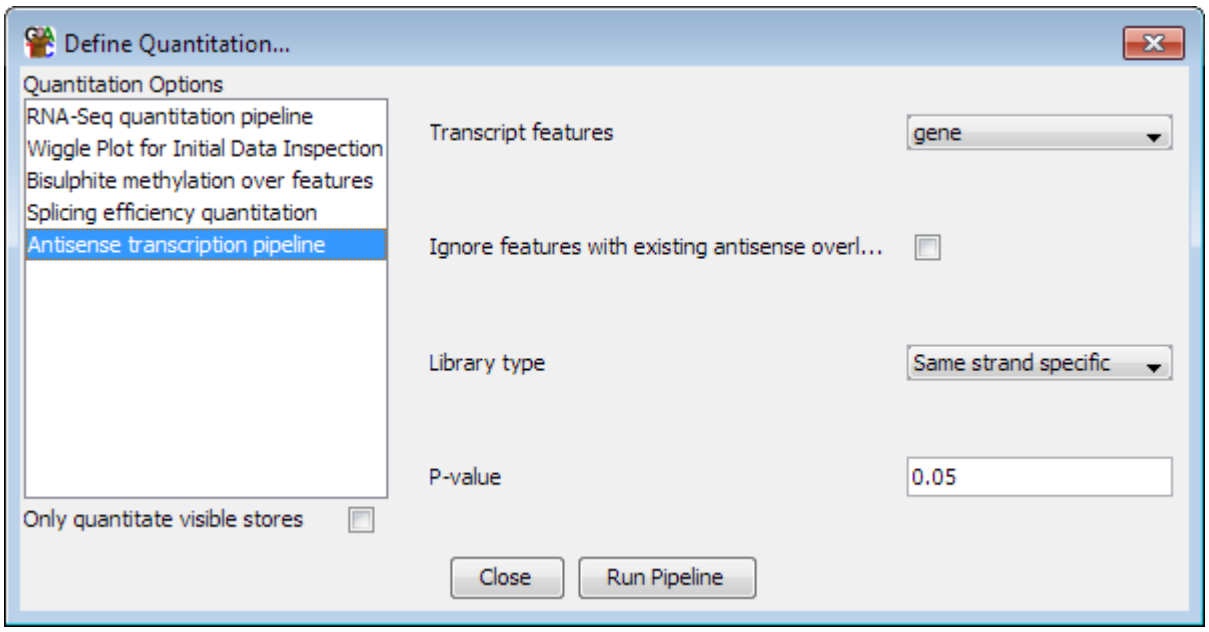
Splicing efficiency quantitation pipeline

The splicing efficiency quantitation pipeline generates a measure of splicing efficiency for RNA-Seq datasets. The basic premise for the pipeline is that samples which exhibit lower splicing efficiency will have a greater proportion of reads in their introns. The pipline therefore quantitates features based on the density of reads in introns and exons. You can then use these values to look for overall

differences in splicing between samples, or to find individual transcripts where this might have changed.

Antisense transcription pipeline

The antisense transcription pipeline again operates on RNA-Seq data, but is specifically designed for data coming from directional libraries. The aim of the pipeline is to identify putative regions which are undergoing antisense transcription.



The pipeline performs both a quantitation and a statistical analysis of a set of genes. It first looks for the genome wide level of antisense transcription to get an idea of how 'leaky' the strand specificity is across the whole library. In a second pass it then analyses each gene individually to see how many reads were found on the sense and antisense strands. It uses a binomial test to see if the number of antisense reads is unexpectedly high, and quantitates the gene with an obs/exp value. As well as producing a quantitation the pipeline also makes a probe list for each sample listing the significantly antisense genes it identified.

Normalisation

SeqMonk provides a series of tools to quantitate your raw data. These include some normalisation options allowing you to correct for factors such as the total number of reads in your dataset and the length of the probe you're quantitating. Although this initial quantitation can provide useful data, in many cases it is susceptible to other sources of bias which might cause systematic differences between your datasets, and which might lead you to make incorrect conclusions about the differences between your data.

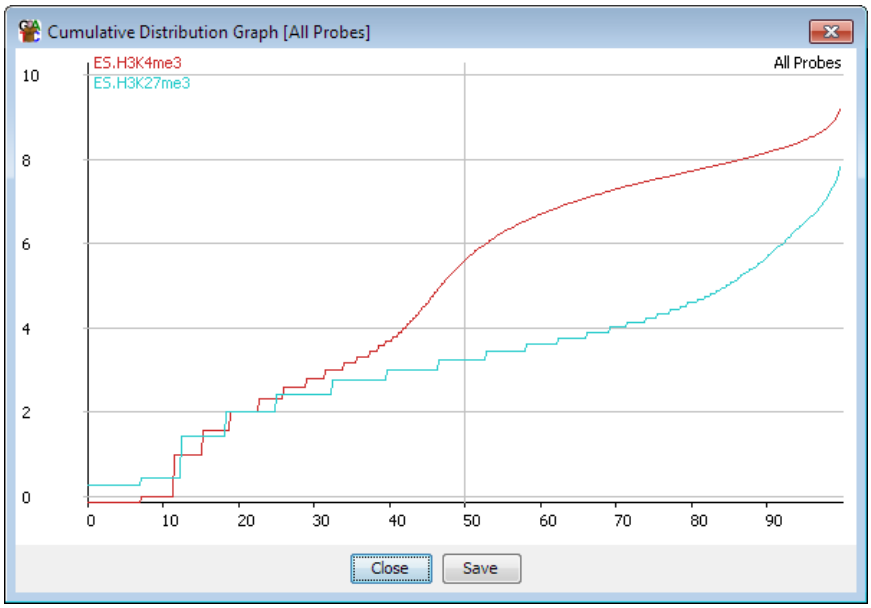
Common sources of bias might be:

- Having greatly different numbers of reads between samples, meaning that low values may be measured with very different accuracies between data sets.
- Having different levels of PCR duplication between samples
- Having different read lengths
- Having different total amounts of signal between samples (eg RNA-Seq samples where there is more transcription in one sample than another.
- Having different degrees of mis-mapping contaminating sequences into different samples

Before you trust your quantitation you should therefore take some time to look at the set of quantitated values you have produced and compare these between your datasets. If there are systematic differences between your samples then you either need to think of why this might make sense biologically, or if you decide the differences are technical you can try to normalise the data so that their influence is removed.

Cumulative distribution plot

The easiest way to visualise and compare the distribution of values you have between your datasets is to use the cumulative distribution plot. This plot simply shows you the path your quantitated data takes to get from the lowest value in your set to the highest. Different datasets which show the same distribution of values (even if individual measures show large changes) should have virtually identical paths on this plot, and the aim of normalising data is to make the paths as similar as possible.



At the low end of this plot you will see some instability and stepping of the plot due to very low absolute counts. In some data sets you will clearly see separate points for probes with 1, 2, 3 etc reads in them. As the absolute counts increase the plot will tend to smooth out, but this may happen at different points in different datasets depending on their coverage levels.

Empty Values

One of the biggest problems when dealing with count based data are places where you saw no data at all. The problem is that because you saw nothing you can have no idea how much more sequencing you would have had to have performed until you saw some data in that position. Comparing an empty probe to one with a small amount of data (especially when there is a difference in the total amount of data collected) is very problematic, and one of the largest causes of false predictions in this type of data.

The problem of empty values is exacerbated when the quantitation values are log transformed. Since you can't log transform a zero value you have to assign some arbitrary value to it to allow it to have a real value after transformation. The problem is further compounded by adding corrections for length and total counts, and at what stage the small value is added to allow log transformation.

In the end the compromise made by SeqMonk is that if count data is to be log transformed then any empty values are given a raw count of 0.9. This applies to both the read count quantitation where 0.9 of a read is added, and to the base pair quantitation where 0.9 of a base is added. Any transformations for probe length and total count are then applied on top of this initial value.

In practice this has some consequences for empty probes in your data.

1. If you log transform and correct for probe length then your empty probes will have different absolute values within a dataset. This makes some kind of common sense since finding no reads in a probe which is 10kb long should not necessarily be viewed the same as finding no reads in a probe with is 10bp long. In extreme cases you may observe a small secondary distribution of empty probes to the left of your probe value distribution, but mostly these will tend to blend in with the bottom end of the very lowly measured reads.

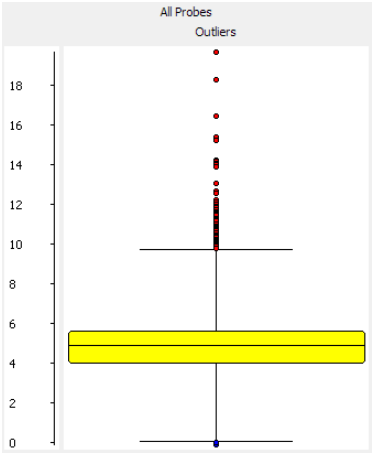
2. If you log transform your data and correct for total read count then empty probes will not have exactly the same value between datasets. The values will be consistent within a dataset (if you haven't corrected for probe length), but you may incorrectly think you have real changes occurring between probes which are empty in both sets if you aren't careful about your filtering criteria.

Whilst this way of handling empty reads isn't absolutely ideal it is the best compromise we can come up with, and in most cases will produce sensible results. The only condition where this will give very misleading results will be cases where you have very large differences in the total number of reads between datasets such that the proportion of empty probes varies wildly. Even in these cases the other normalisation techniques described here should identify and help to correct this problem, but in general you need to stay aware of the accuracy with which your measurements are being made.

Removing outliers

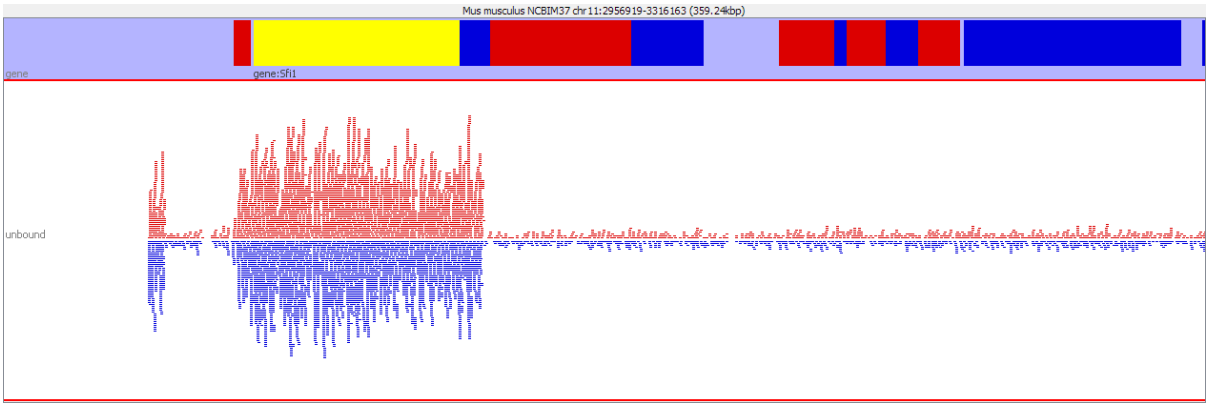
One obvious source of bias is the presence of outliers in your data. Many of the quantitation methods used in NGS analysis apply some kind of correction for the total number of sequences in the dataset. The assumption made is that the general distribution of sequences over the genome is similar between datasets and that you can scale the actual counts you got up or down to match the total number of sequences in each dataset.

There are a few problems with this assumption. One of the major problems is that we tend to assume that all of the genome is represented in our current genome assembly. Whilst many genomes have very good assemblies they still have large holes over specific regions, and these can cause problems. In particular telomeric and centromeric regions are very large (and indeed variable in length), and are almost completely unrepresented in the assemblies due to their highly repetitive nature. However sequences from these regions will still turn up in our libraries. This wouldn't be a problem if our mapping programs caused them never to be mapped to the genome, but what we see instead is that some proportion of this extra sequence is mapped incorrectly to a position within the genome assembly. These incorrect mapping positions then show huge enrichment, even in unbiased or control samples, and the huge number of counts from these mismapped sequences can have an influence on the global count corrections which are applied. In the plot below, which shows read counts on a log scale – the single top outlier covers only 1:500000th of the genome, but contains 4.5% of all of the reads.



As an example of this, if you have a look at the distribution of read counts over a genome you will tend to see a strong upward spike in read counts at the ends of many of the chromosomes. This spike results from incorrectly mapped reads coming from the repetitive sequence which isn't part of the

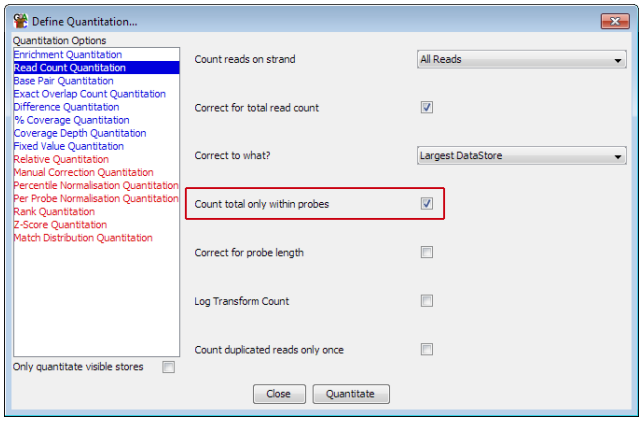
assembled genome. The classic example of where you see this is the Sfi1 gene in human/mouse which has many of these repeats in it, and consequently is often predicted as having a significant biological effect, when actually it's a victim of incorrect normalisation.



In addition to the generic problem of repeats, some techniques will specifically enrich for a specific class of repeats, and an increased mismapping ratio for these sequences will produce a global bias in the normalisation. MeDIP samples, for example, often enrich major satellite sequences which can comprise up to 40% of all sequences in the library. These are generally absent from the assembly, and can thus produce a small number of strong mismapped peaks, which can contain up to 20% of all mapped sequences.

One way around this problem is to simply remove extreme outliers before performing your final quantitation. The outliers we're talking about here will have an enrichment level way in excess of anything which could be produced by a ChIP or RNA-Seq enrichment and could only realistically be derived from mapping artefacts. Typically they will have read counts two or three orders of magnitude above the rest of the distribution and can be removed by a simple values filter. If you have an input sample which you expect to show an even distribution then you can use the box-whisker filter with a harsh filter to find these mismapping events globally and remove them.

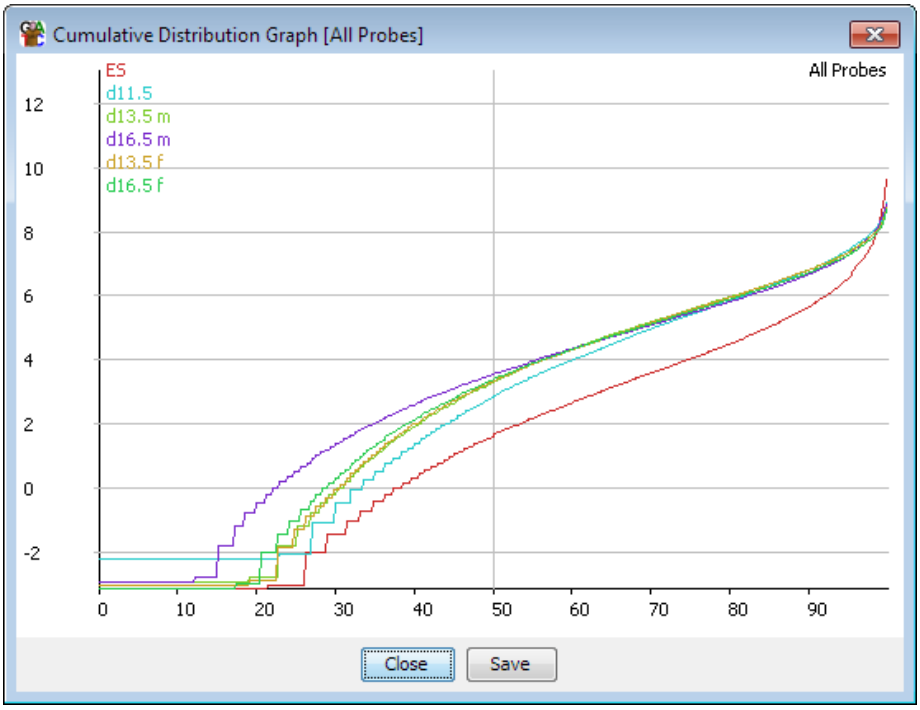
Once you have filtered your original probe list to remove these aberrant regions then you need to use the 'Existing Probe List Probe Generator' to promote the filtered list up to being a full probe set which you can then re-quantitate. When re-quantitating you need to ensure that you only count reads within the probe set when correcting for total read count.



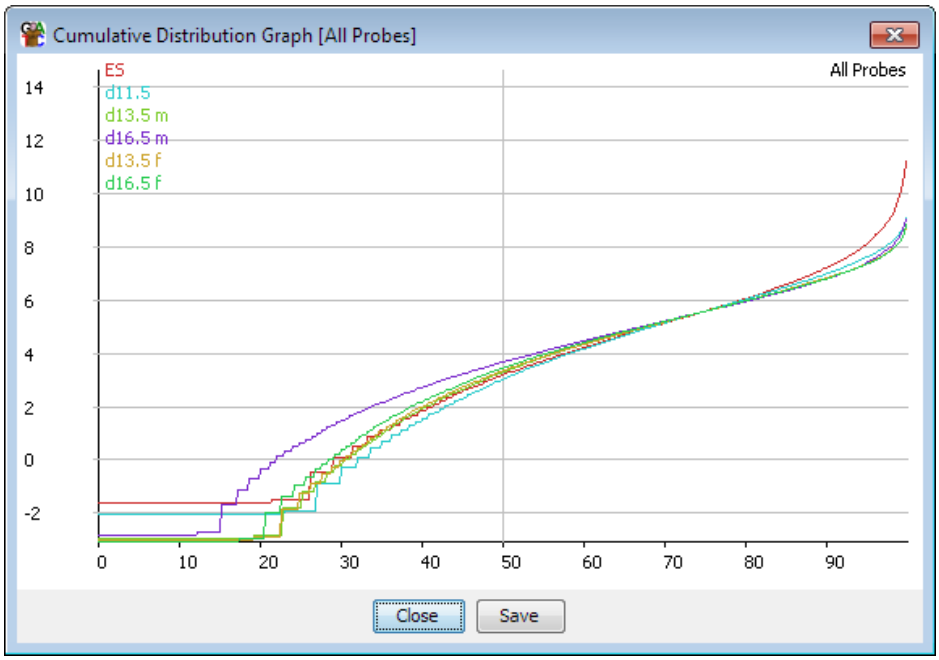
Percentile normalisation

A commonly observed pattern, especially on ChIP and other enrichment based libraries, is that your distributions follow a similar path, but on a somewhat different scale. Thus the original quantitation would suggest that there is a consistent difference between your samples, and scatterplots would show an off-diagonal consistent relationship between the samples. This kind of discrepancy can be caused by a change in the proportion of reads falling into the probes being measured, either through mismapping, or through a change in the efficiency of the ChIP enrichment. Since the differences observed do not in most cases represent an actual biological change then it is reasonable to aim to normalise away this difference.

The simplest tool to achieve this type of normalisation is the Percentile Normalisation Quantitation. This quantitation method allows you to set a reference percentile in your data and the existing quantitations will have a correction factor applied to them such that their values at your chosen percentile will match exactly. Normally it makes sense to set the reference percentile to somewhere around 75% since this will be in the well measured portion of your data, but before any big changes which might occur at the extreme end of the distribution.



You can also choose what kind of correction will be applied. The default is an additive correction where a constant value is added to each point to get the distributions to match. The downside to this method is that the same correction will be applied everywhere, and will end up with different values being set for the empty probes in your different data sets. A more appropriate correction in these cases is to use a multiplying factor to correct your data. This will 'stretch' your distribution to match at the specified percentile and may more closely match the overall distributions, especially at the low end.



Since the same correction is applied to all points on your distribution this is a fairly safe and uncontentious correction to apply, but in some cases this correction alone will not be enough to completely match the distributions of your samples.

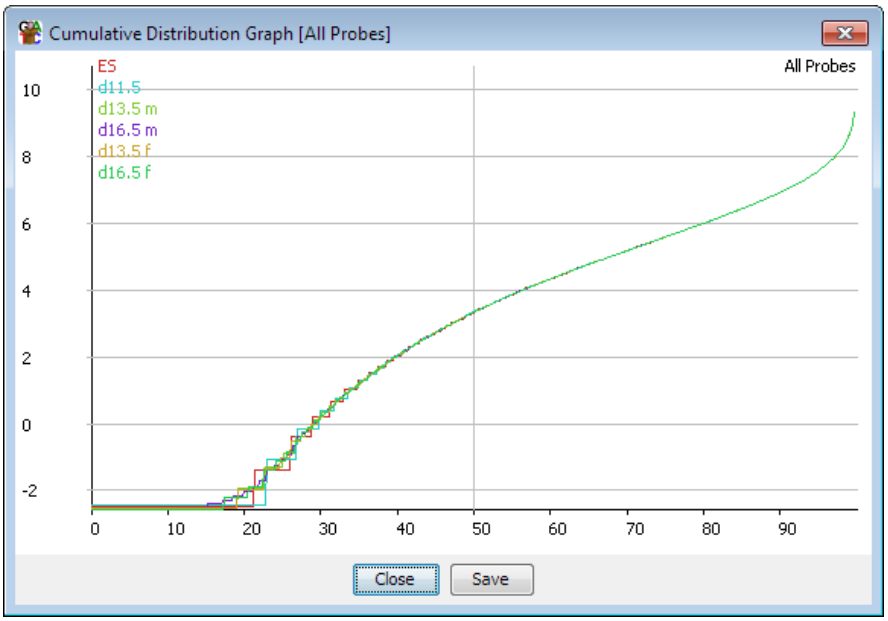
Matching distributions

If you have found some variation in the distributions of your samples which you are unable to satisfactorily remove using the total read correction, length correction or percentile normalisation correction then the ultimate way of making your distributions match is to use the Match Distribution quantitation method. This method makes up an averaged distribution from all of your current samples, and then forces all of your individual distributions to follow this average distribution exactly. You will therefore end up with perfectly matching distributions (or at least as close as your measurement resolution will allow – the method will not cause probes which previously had the same value to have different values).

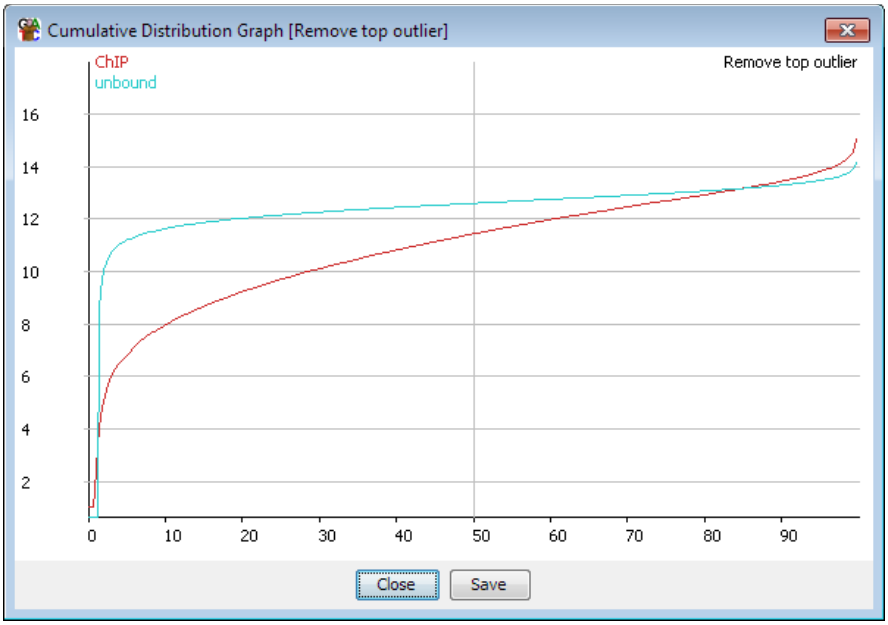
You can think of this method of normalisation as being similar to the 'Convert to ranks' normalisation method, which simply removes all of the quantitation from your data, and gives a value to each probe simply based on its position between the lowest and highest values. In this case we do something similar, but instead of putting the values on a straight line from lowest to highest, we follow the average trajectory which the original values took, thereby preserving any gross features in the distribution.

Performing this kind of normalisation is more risky than something like the percentile normalisation because it does not treat every probe equally. It forces all samples onto the same distribution whether or not that makes biological sense. You therefore need to be sure that you're definitely not interested in whatever differences you are using this method to remove. For example it would be completely inappropriate to normalise both the input and ChIP samples from a ChIP-seq experiment this way, since the input would be expected to show a much flatter distribution than the enriched sample, however normalising several ChIPs from the same antibody might be more justified. In other samples it might be valid to use this to normalise RNA-Seq samples which had different coverage and had then had to be deduplicated, since these will show an intensity dependent bias, but it would be bad to

use this to match up distributions between RNA-Seq samples which had different overall amounts of transcription.



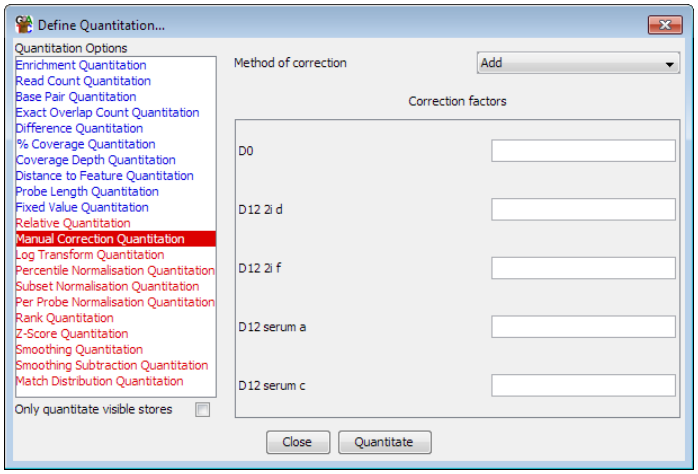
It would also be bad to use this on enrichment samples where there was truly a biological reason for the different amount of enrichment. An extreme example of this would be the case of a ChIP vs and input sample, where you'd expect the input to be much flatter, and the ChIP to show a wider dynamic range.



In any case, the match distribution normalisation should be the last step in your normalisation protocol, and you should try to match your distributions as closely as possible using the other methods mentioned before using this, since otherwise some samples will have much more influence over your averaged distribution than others.

Manual normalisation

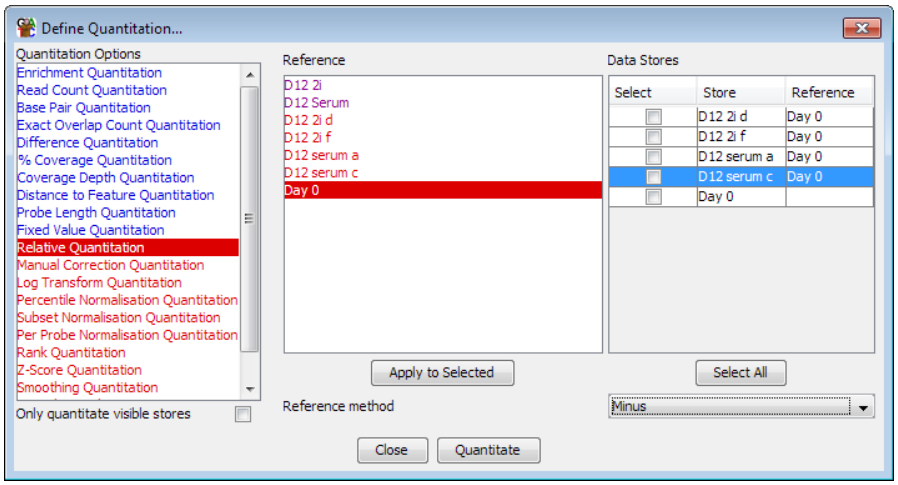
For some specialised applications it may be that the information you need to correctly normalise your data isn't present in the mapped sequences themselves. Most sequencing applications are by their nature relative measures, telling you what proportion of reads fall into a particular gene, but not how this relates to an absolute measure in your original sample. To take a simple example, two RNA-Seq datasets taken from samples where the distribution of transcripts were identical, but the absolute level of transcription was different, would look exactly the same in the mapped data. Only by incorporating some external measure could you account for this type of difference.



To allow you to apply these kinds of correction SeqMonk has a manual correction option. This lets you enter a manual correction value for each of your datasets and choose how this is applied to the set of quantitated values calculated by SeqMonk. By using this option you can incorporate absolute external measures with the relative measures the program itself can produce.

Normalising to other samples

In some cases you might want to normalise your samples against each other - quantitating sample A as the difference between sample A and sample B for example. There is a quantitation module which lets you apply this type of more complex normalisation called the relative quantitation module.



This module allows you to pair up your samples and then choose what operation to apply between each sample and its selected reference. To match a sample and reference you select the reference from the reference list and then check the boxes next to the samples to which you want to apply this reference. Presssing the “Apply to selected” button will then assign that reference to those samples. Once you’ve paired up all of the samples you want to correct you can choose how you want to normalise (subtract reference, divide by reference etc) from the drop down box at the bottom.

A lot of people are keen to use this type of quantitation for enrichment type experiments such as ChIP-Seq and would use it to normalise against an input sample. Whilst this is a common procedure, you should be somewhat cautious about applying it. When you normalise against a reference the final accuracy of your normalised value is dependent on the accuracy of both the enriched and the reference samples. In a ChIP experiment the enriched sample is generally measured very well since the reads will cluster in the measured regions, but the input sample is often poorly measured since the reads are evenly spread over the whole genome. This means that the normalised values are more likely to be influenced by technical variation in the input measures and that normalising may make your data worse. As an alternative you could consider using the input samples as a filter, to remove places where the input shows significant enrichment, and which are therefore also likely to give inaccurate measures in the enriched sample.

Statistical Tests

SeqMonk includes a few different statistical tests which can be used to more rigorously explore your data. In general these tests fall into two classes – those which require biological repeats to have been performed, and those which can operate on single replicates. A true statistical analysis of all sources of variation in your data will require biological replicates, so this type of experimental design is preferred if at all possible. However sequencing experiments are still expensive, and samples are sometimes limited, such that it becomes necessary to analyse single replicates, and to use statistical tests to account for technical rather than biological sources of noise.

Subgroup Significance Test

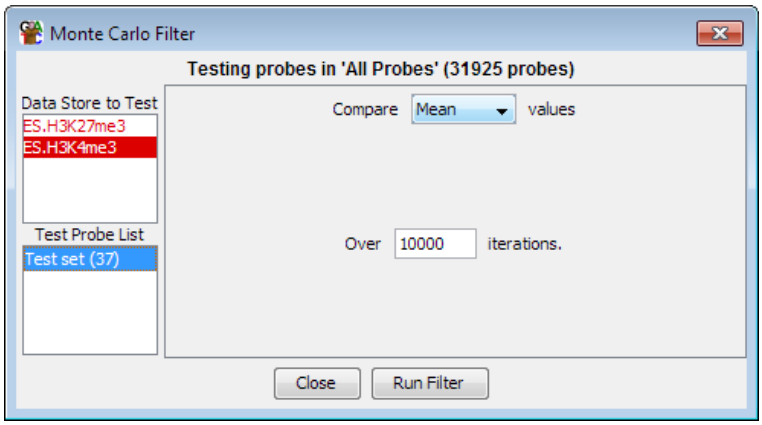
One of the tests people often want to perform is to see whether a selected subgroup of probes shows unusual properties, or whether a similar randomly selected group could show the same features. The easiest way to assess this is to perform a simulation where many random groups are selected from the same starting set, and then compared with the original group. You can then assign a p-value based on the frequency with which the random groups showed a quantitation equal or higher to the initially selected group.

This type of test is called a Monte-Carlo simulation, and SeqMonk allows you to perform this kind of test on your probe lists.

Monte-Carlo Simulation

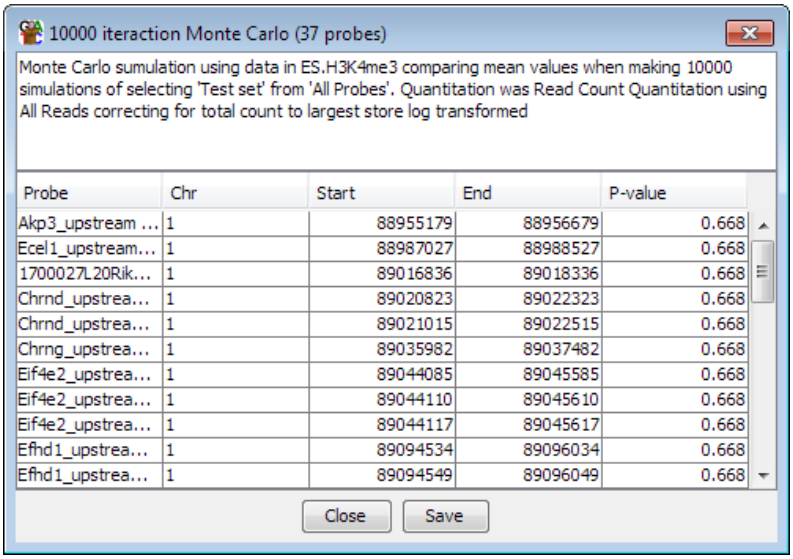
The basis for a Monte-Carlo simulation in SeqMonk is two sets of probes, where one is a subset of the other, and a data store whose quantitated values will be used for the test.

You start by selecting the parent group, from within which the randomly selected groups will be drawn. In the options for the tool you will then see all of the children of that group, from which you can select the subset you'd like to test. You can also select which data store you want to test on.



You can then select which quantitation measure you'd like to use for the test, you can pick either the median, mean or maximum value from the set of probes, and then the number of iterations you want to perform. A Monte-Carlo simulation doesn't produce an exact result – the number of iterations selects how close to the true p-value you want to get. Performing more iterations will increase the accuracy of the p-value and will lower the lower bound of the result. The default is 10,000 tests which is pretty sensible and allows for a lower p-value limit of 0.0001.

Once the test has run then a probe list will be created as a child of the test list which will always contain exactly the same list of probes as the test list itself. The only difference will be that each probe in this list will be annotated with the p-value from the simulation.



Probe	Chr	Start	End	P-value
Akp3_upstream ...	1	88955179	88956679	0.668
Ecel1_upstream...	1	88987027	88988527	0.668
1700027L20Rik...	1	89016836	89018336	0.668
Chrnd_upstrea...	1	89020823	89022323	0.668
Chrnd_upstrea...	1	89021015	89022515	0.668
Chrng_upstrea...	1	89035982	89037482	0.668
Eif4e2_upstrea...	1	89044085	89045585	0.668
Eif4e2_upstrea...	1	89044110	89045610	0.668
Eif4e2_upstrea...	1	89044117	89045617	0.668
Efhd1_upstrea...	1	89094534	89096034	0.668
Efhd1_upstrea...	1	89094549	89096049	0.668

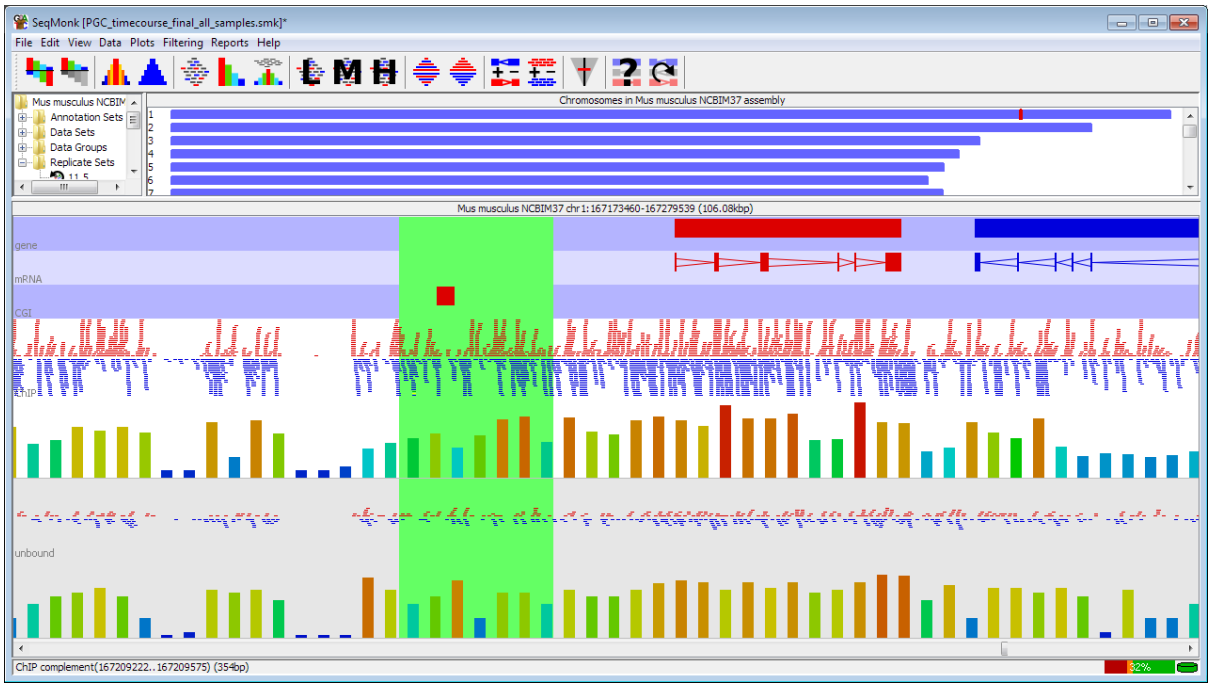
Single Replicate Tests

SeqMonk has two statistical tests suitable for the analysis of single replicate data. These are the windowed replicate test and the intensity difference test.

The Windowed Replicate Test

The windowed replicate test is an implementation of a standard T-test (1 or 2 samples) or ANOVA (more than 2 samples). Since it is intended for use on non-replicated data the way it acquires a set of values for each condition is to analyse a set of probes which are located physically close together. The test allows you to specify a set of windows, which must be large enough such that several probes are likely to be contained within each window. The windows can either be fixed size windows which are slid over the genome, they can be fixed numbers of adjacent probes, or they can be defined by a feature type, with a test being performed for the set of probes which fall under each feature of that type.

The test then takes this set of probes and treats them as it they are technical replicates of the same biological effect. It can then test the values for this set of probes against other samples an assign p-values based on the differences. Probes pass or fail the test as a windowed set. If a particular probe passes the test in any window then it will be kept in the filtered list, regardless of how many other windows failed the test.

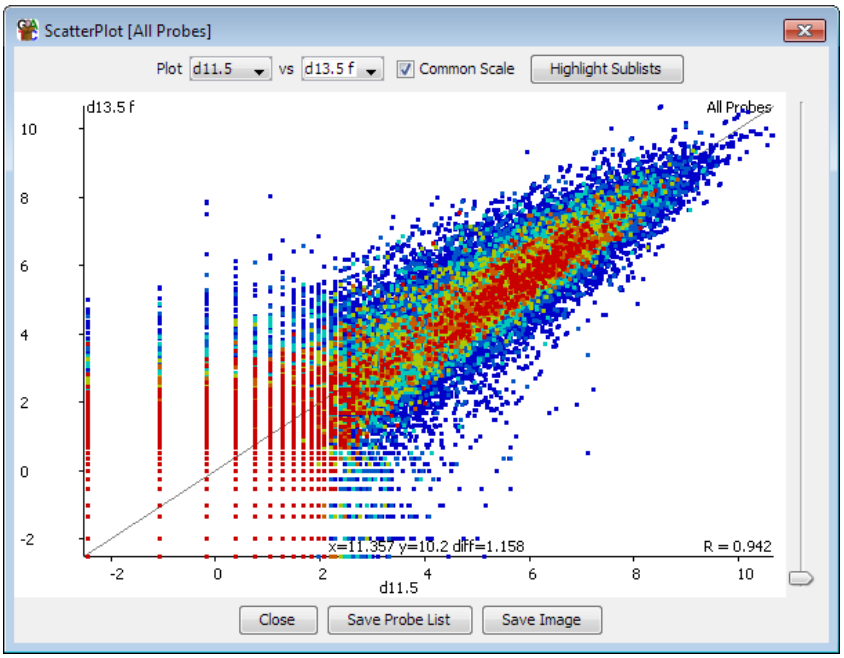


In the above example a typical test would be represented by taking the set of 7 probes contained within the highlighted area (representing the chosen window size) and using a t-test to compare their distribution between the two samples. If the test passed then all 7 probes would be put into the pass list, even if the same probe failed in a different window.

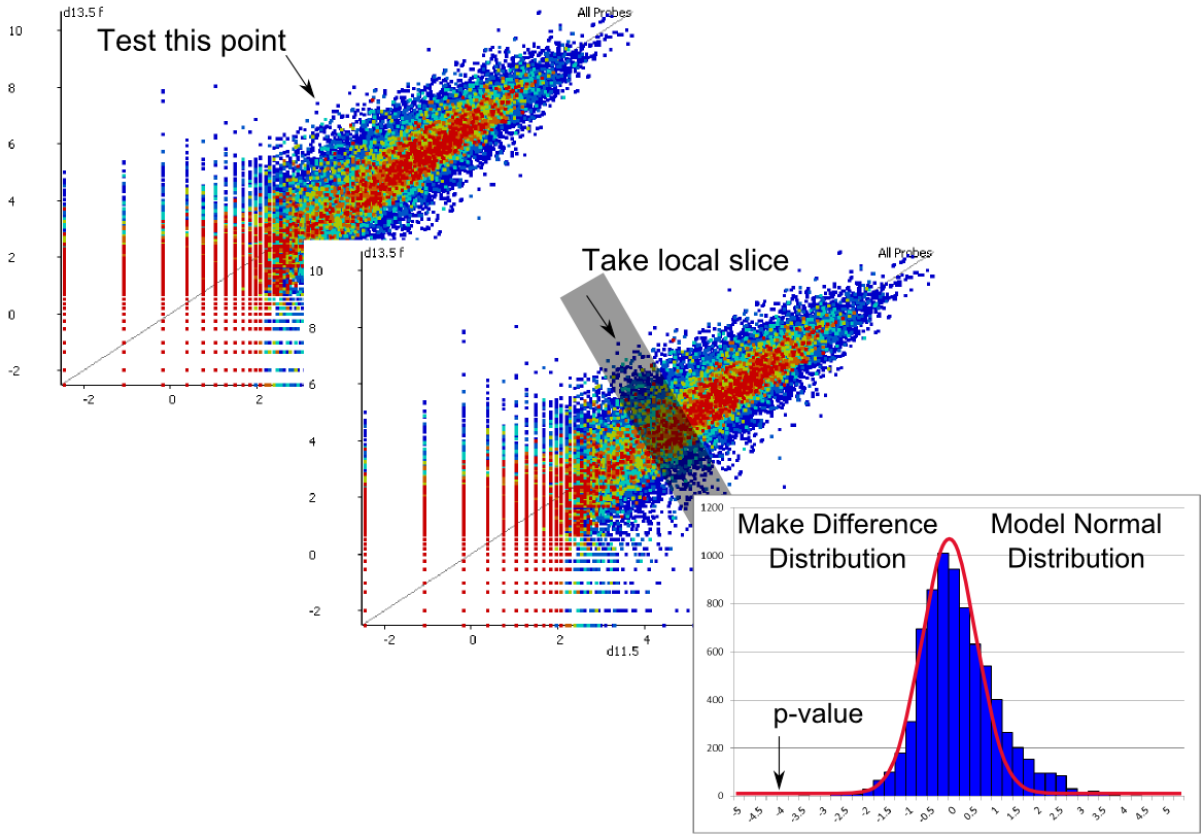
This test is useful where you expect to see a consistent effect across several closely related probes, but doesn't tell you anything about biological variation. It could be used to test, for example, whether the set of methylation calls underlying a feature (a CpG island for instance) showed a consistent difference between two or more samples.

The Intensity Difference Test

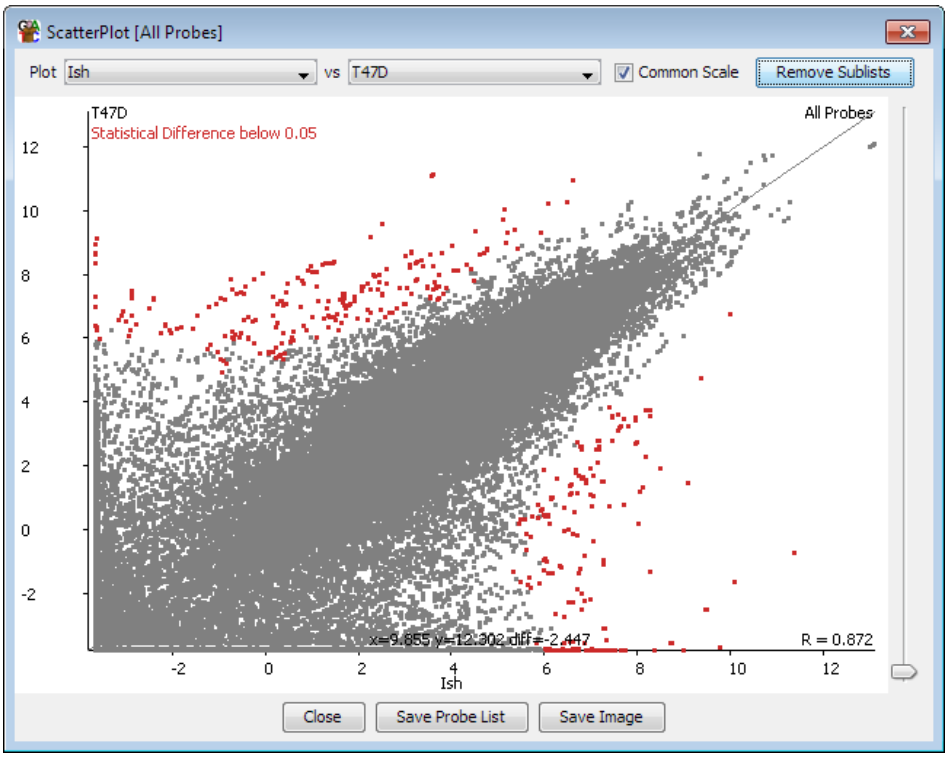
The intensity difference test performs a statistical test on your data using the general distribution of your data to test whether any individual point is likely to be an outlier from the overall distribution. It is a pairwise test, but can be extended to multiple samples by doing a series of pairwise tests and combining the results. The test is useful for cases where you have an intensity dependent level of noise in your data, such that a simple fold change or z-score cutoff is not an appropriate way to find outliers. The test is only valid for datasets where the majority of points do not change, and you are only interested in a relatively small proportion of outliers.



The test works by taking each point in turn and extracting from the pair of datasets being examined a subset (normally 2% of the data) of points with the closest average intensity to the point being examined. From this set the program then constructs a distribution of the differences between your two datasets. This distribution is then modelled against a normal distribution, allowing the calculation of a specific p-value for the likelihood of finding a point as far away from the centre as the point you are testing.



After correcting for the number of tests performed you then have an intensity dependent test which operates without replicates on each probe individually.



This test is widely applicable for ChIP-seq, RNA-Seq and other types of data, where data exhibits intensity dependent variability, and where most of the probes are not changing between conditions.

You can also use this test on datasets where you have biological replicates by putting the replicates into a replicate set and then comparing two replicate sets with this test. You can use this test to identify probes which have an average difference which places them outside the general level of disruption seen in your system, and then use the more conventional replicate tests to test whether the difference identified are consistent across your replicates. This gets you a relatively small number of probes which are likely to represent the most biologically interesting differences in your data.

Multiple Replicate Tests

SeqMonk currently contains a single test for use on multiple samples. Use of this statistical test requires that you have grouped your biological replicates into replicate sets so the program knows which groups of samples if can compare. At least 3 individual samples must be included in a replicate set in order to allow it to be used for statistical analysis, although larger numbers of replicates will provide more analytical power.

The test used for comparing replicate sets is the Replicate Set Stats test. The actual statistical test performed will depend on the number of replicate sets you select. If only one set is selected then a single factor T-test is performed and your points are compared to a fixed value of 0. For two samples a normal unpaired, 2-tailed T-test is used, and for more than two sets an ANOVA is performed.

Because this type of test can use the variation between replicates as a measure of both biological and technical noise the test can be applied to a wide range of different data and quantitation types. The

only real assumption made by the test is that the noise in your system is generally normally distributed, which should be true in most cases.

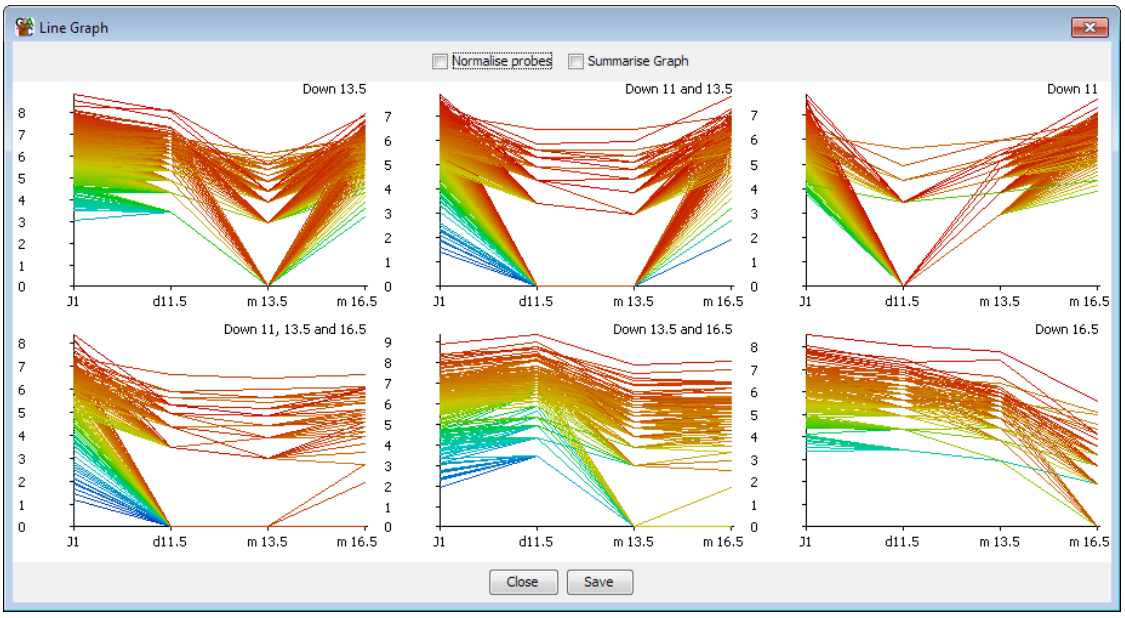
Clustering

As experimental designs become more complicated it can increasingly be the case that simply generating lists of probes which are changing in your experiment is not very useful, since this list may well contain subgroups with clearly different patterns of change. In a simple pairwise comparison you can split changing probes into those which go up and those which go down, but as the number of experimental conditions increases so does the number of possible profiles of changing probes.

One way to approach this problem is to perform a clustering on your initial list of changing probes so that instead of a single list you generate several lists, where the members of each list are all changing in a roughly similar way. You can then start to analyse these sub-groups as a set to try to identify common biological features of the group which might explain the observed pattern of change.

The line graph plot

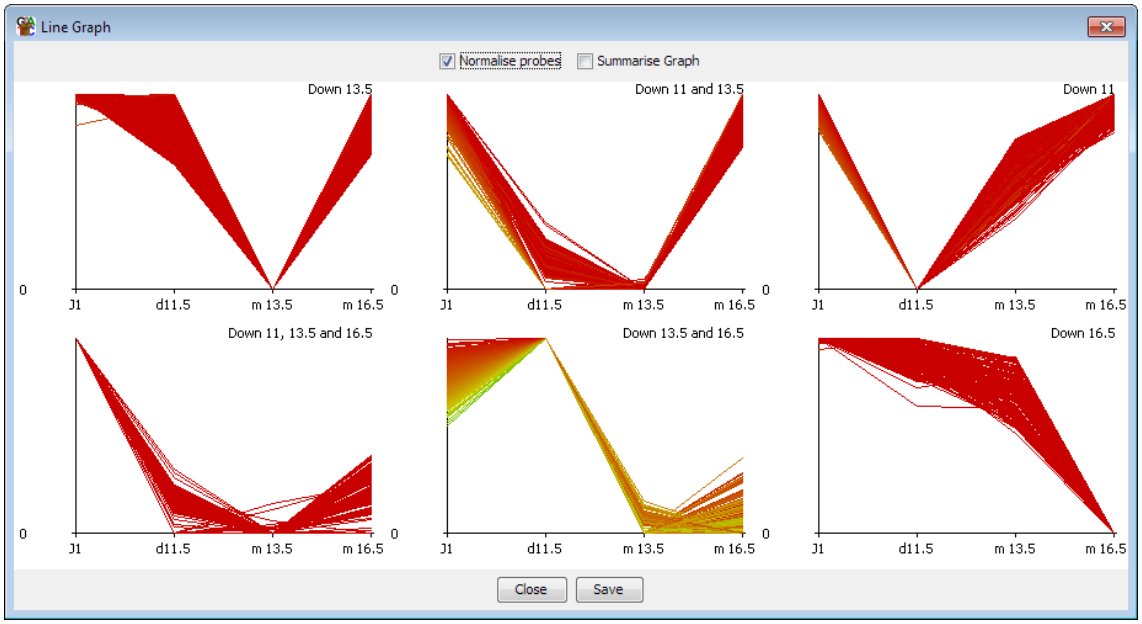
Before we get into doing the actual clustering it's useful to know how the results can be displayed. SeqMonk provides the line graph plot which allows you to look at the changing quantitation of thousands of probes across multiple datasets, and which can also show several groups (clusters) at once.



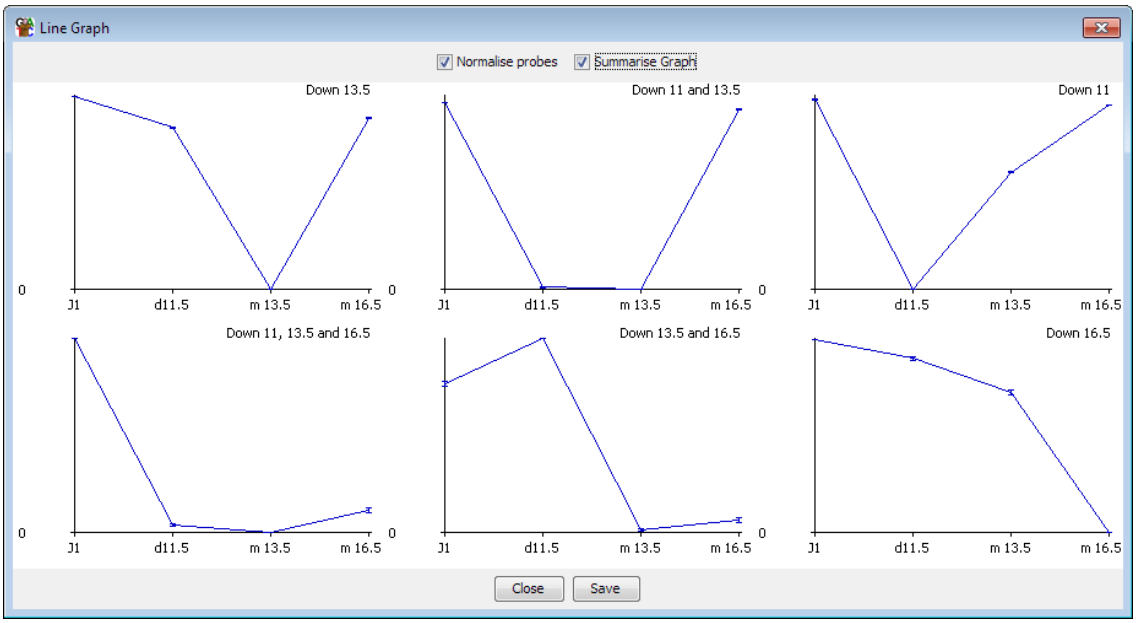
In a clustered set of probes the shape of the lines over the various data stores should be somewhat similar. However, with large numbers of probes you may still find the plots descending into chaos. There are therefore a couple of additional options which can help to clean things up.

Clustering is based on observing similar patterns of change between conditions, however when the same pattern of change is observed at different absolute levels of quantitation you end up with a line graph containing multiple parallel lines. With large numbers of probes these parallel lines merge into a block, such that it is hard to discern the overall pattern observed in the group. One way to clean up the view is therefore to perform a per-probe normalisation. For each probe this normalisation subtracts the median value across all data stores from each individual point. In effect it adjusts the y axis for each probe so that it centres on a value of 0. What you are then comparing is the variation in the quantitated values between conditions, regardless of the absolute level of quantitation. The plot should therefore be more consistent between the different probes and the overall pattern should

emerge more clearly. This per-probe normalisation can be performed in real time in the line graph plot, but should you wish to you can also make the same change to your raw data by using the per-probe normalisation quantitation method.



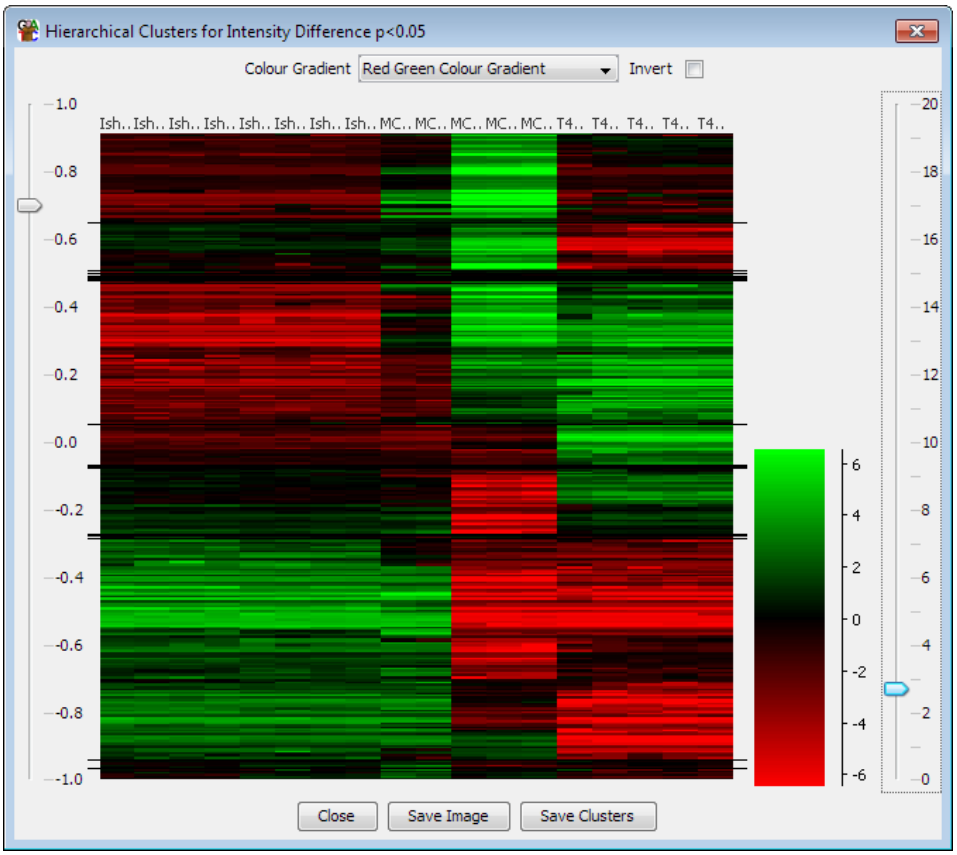
If the plot is still somewhat confused even after per-probe normalisation then a second additional option to clean up the plot is to just show the average value for each dataset, rather than showing each individual point. In this mode a single line is shown, but for each point a set of error bars indicate the standard deviation of the values in that data store. This should provide a good view of the overall pattern across several samples even where there is a high level of variability.



Hierarchical clustering

If you have a relatively small number of probes to cluster (fewer than 5000 would be a sensible limit) the best place to start looking at clustering is the hierarchical clustering tool. This is a visual clustering

where a set of probes is rearranged so that probes with a similar quantitation profile across your set of conditions are placed closely together.



As you can see there are obvious groupings of probes within the plot. If you want to be able to define these groups then you can use the slider on the left to divide the plot into groups which show a level of correlation above a threshold you specify. You can then save these groups as probe lists in your project. You can zoom into the y-axis of the plot by dragging box within the plot area. When you export lists of probes only groups which are currently visible in the plot are exported.

Automated Correlation Clustering

Another place to start working with clusters if you have a large number of probes is to perform an automated correlation clustering. Automated clustering requires at least 3 different data stores to work with and is based on grouping together probes whose quantitation patterns across those stores shows a high Pearson's correlation coefficient. It is implemented as a probe list filter and can be found under Filtering > Filter by Correlation > Filter by Correlation Cluster.

When setting up the clustering you should start with a set of probes you already know to be changing between your conditions. Including unchanging probes will start to cluster on the noise in your system and your results will be poor and take ages to calculate.

The process for automated clustering is that the first probe analysed starts a cluster. Every probe after that is correlated to the existing clusters. If it can be placed in an existing cluster then it is. If not then it gets to start a new cluster. Finally, when all probes have been clustered you can apply a filter to remove clusters with very low numbers of probes in them.

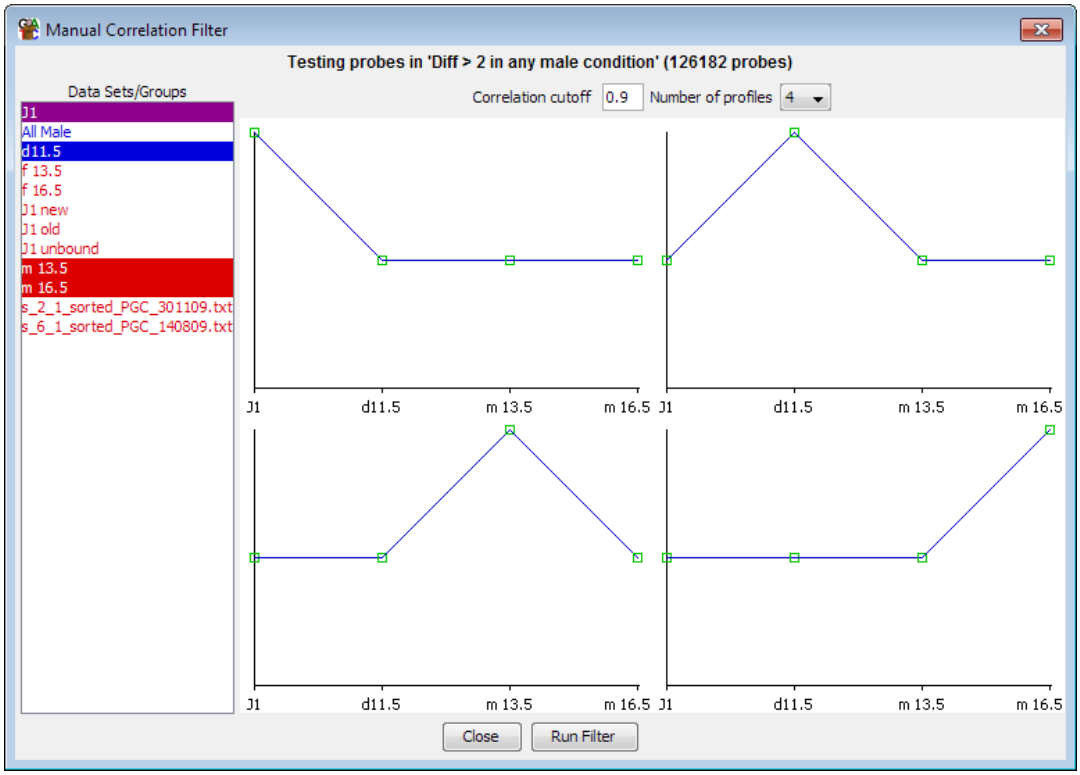
The stringency of clustering is based on the degree of correlation you want to see within each cluster. Setting a high correlation cutoff (eg 0.9) will produce lots of small clusters with very similar profiles. Using a low cutoff (eg 0.7) will produce fewer, larger clusters but with more variability in the profiles contained within an individual cluster.

Automated clustering can be a useful step in analysing your data, but the results it produces will depend on the order in which probes are passed to it. The results can therefore be somewhat noisy – you may sometimes see two or more clusters produced which have very similar overall profiles. The clusters produced by this automated clustering should therefore not be taken as a final answer but will give you an idea of the breakdown of your probes, and once you know this you can use the manual clustering options to get a cleaner set of results for the interesting clusters you find.

Manual clustering

Rather than performing an unbiased automated clustering, the other option you have is to do a manual clustering. In a manual clustering you define a starting set of profiles you want to find. Every probe is then correlated against this set of profiles – any probe which fails to correlate to any of the profiles is rejected and those that do correlate are placed in a group with the profile to which they correlated most tightly.

Unlike automated clustering which can give noisy and inconsistent results, the results of manual clustering should be cleaner and reproducible, but you will only find the profiles you specified to start with.



Profiles for manual clustering can be specified in one of two ways. You can choose to create a profile manually, in which case you are presented with a list of your currently visible stores and you can manually drag the control points on the profile shown to create the profile you want to see. This gives you complete control over the shape of the profile you want to find. Alternatively you can select an existing probe list and the clustering will create an averaged profile from the data in this list and then

cluster against this profile. The lists you select can contain any number of probes, so you could, for example, make a list containing a single probe if you wanted to find other probes which were similar to a probe of interest.

Tips and Tricks

Working with Annotations

Each SeqMonk genome comes with a core set of annotations derived from the Ensembl project, however you are free to bring in as many additional tracks of annotation as you wish using the tools under File > Import Annotation.

Importing Annotation

Annotation can be imported from standard annotation format files (GFF, GTF) or from arbitrary text files. Annotations imported from text files are limited to only having a single location (so not allowing split locations such as a transcript made up of exons), and having only a name and a description. If you want to use more advanced options such as split locations and multiple annotation tag-value pairs then you will need to create GFF/GTF files.

One aspect of annotation import which has caused some confusion in the past is the naming of annotation tracks. A single imported file can contain information for multiple different annotation types (genes, mRNA, CDS etc). If no type is specified, by picking a type column during import or manually entering a type, then all of the features are given a type which matches the name of the file from which they came.

One a file has been imported you can change the name of the annotation set in the data view. This will only change the label on the data view and won't actually rename any of the features. If you want to change the name of the features in an annotation set then you need to go through the data view and select Rename Feature, then select the feature type you want to rename (there may be only one) and then give it a new name.

In the chromosome view features are grouped by their type, and no distinction is made between features coming from different annotation sets. If you import a load of new features with type 'gene' then these will be merged in with the gene features coming from the core genome. The only difference will be that you can delete annotation sets from within the Data Panel.

If you import a GTF file but want to distinguish the gene and transcript features from those in the core genome then one extra option during GTF import is the ability to add a specified prefix to every feature type created, so that you could have, for example, flybase_gene alongside the normal gene track.

Probe Lists into Feature tracks

Whenever you run a probe generator or a quantitation pipeline then your current set of probes, including any filtered lists you may have generated will be deleted. This can be inconvenient if you wanted to preserve the positions of a list which you had created. Whilst it is not possible to have multiple probe sets in the same project, what you can do is to transform a probe list into an annotation track. Since annotations are not affected by changing probe sets the positions in your list will therefore be preserved in the project and you can subsequently filter against these using the feature filter tool.

To convert a probe list into a feature track you select File > Import Annotation > Active Probe List

Feature searches into Feature tracks

In addition to their name and description the core features have lots of different pieces of annotation in them. You can search through this annotation using the standard search tool and this can produce interesting lists of features which share a common property (Gene Ontology category, link to other dataset, biotype etc). Rather than just viewing these search results interactively you can choose to turn your list of hits into an annotation track so that they can be visualised over your whole genome, and so that you can subsequently use these for probe generation, filtering or reporting.

As a further filter you can also choose to select specific lines from within a feature search and use just these to create your annotation track. You can combine the results of multiple searches by simply giving the features you generate the same name for their type. All features of the same type will appear in a merged track, however many import events they come from.

Classifying Genes

In more recent Ensembl releases an increasingly large number of different types of gene and transcript prediction have been added in. The gene and transcript tracks now include entries known to be pseudogenes and those which should be degraded by nonsense mediated decay. What this means in practice is that if you are trying to do an analysis across all transcripts your results may be being diluted by including entries which you would not normally consider to be a valid transcript.

To try to help with this problem more recently updated genomes in SeqMonk have started to have the 'biotype' tag added to their gene and transcript features. This lists the sub-classification which Ensembl have applied to each of their gene models, and which will allow you to identify only the more conservative models to use for your analysis.

Although all biotypes are mixed together in the default annotation tracks you can use the feature search tool to generate a track containing just the sub-type in which you are interest. For example you could do a search for 'protein_coding' in the mRNA track, and then save the hits to a new track called mRNA_coding which you could then use for some downstream analysis. For reference, the current list of biotypes used by Ensembl in the mouse genome is below (listed in order of prevalence). The ones with stars next to them are already separated out into separate feature tracks by SeqMonk.

- protein_coding
- processed_transcript
- retained_intron
- nonsense_mediated_decay
- transcribed_processed_pseudogene
- lincRNA
- antisense
- *miRNA
- *snoRNA
- *snRNA
- transcribed_unprocessed_pseudogene
- pseudogene
- *misc_RNA
- *IG_V_gene
- *rRNA
- Retrotransposed
- unprocessed_pseudogene

- processed_pseudogene
- *IG_J_gene
- sense_intronic
- non_coding
- ambiguous_orf
- *IG_D_gene
- *Mt_tRNA
- *IG_C_gene
- polymorphic_pseudogene
- unitary_pseudogene
- TEC
- ncrna_host
- *Mt_rRNA
- disrupted_domain

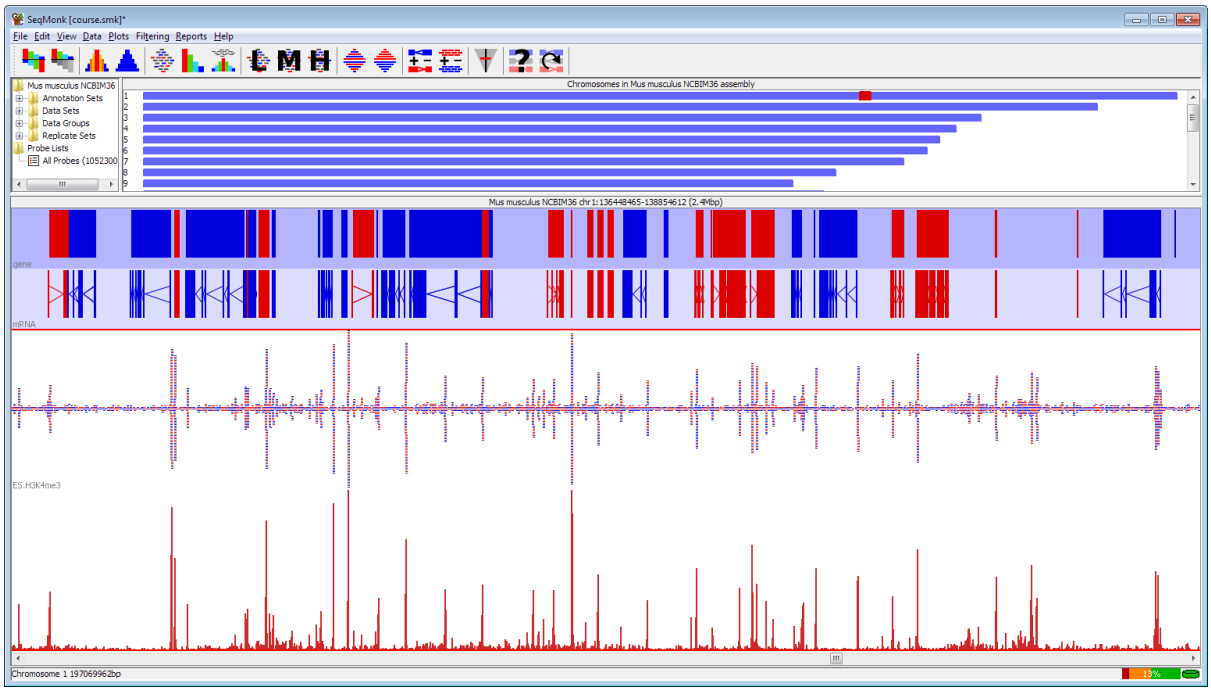
Raw Enrichment Views

One of the views people often like to see of their data is an unbiased enrichment view which allows you to look at the overall distribution of reads in a properly quantitative way. In older versions of SeqMonk this had always been problematic because it didn't cope well with having very large numbers of probes, and the chromosome display wasn't optimised for this kind of view.

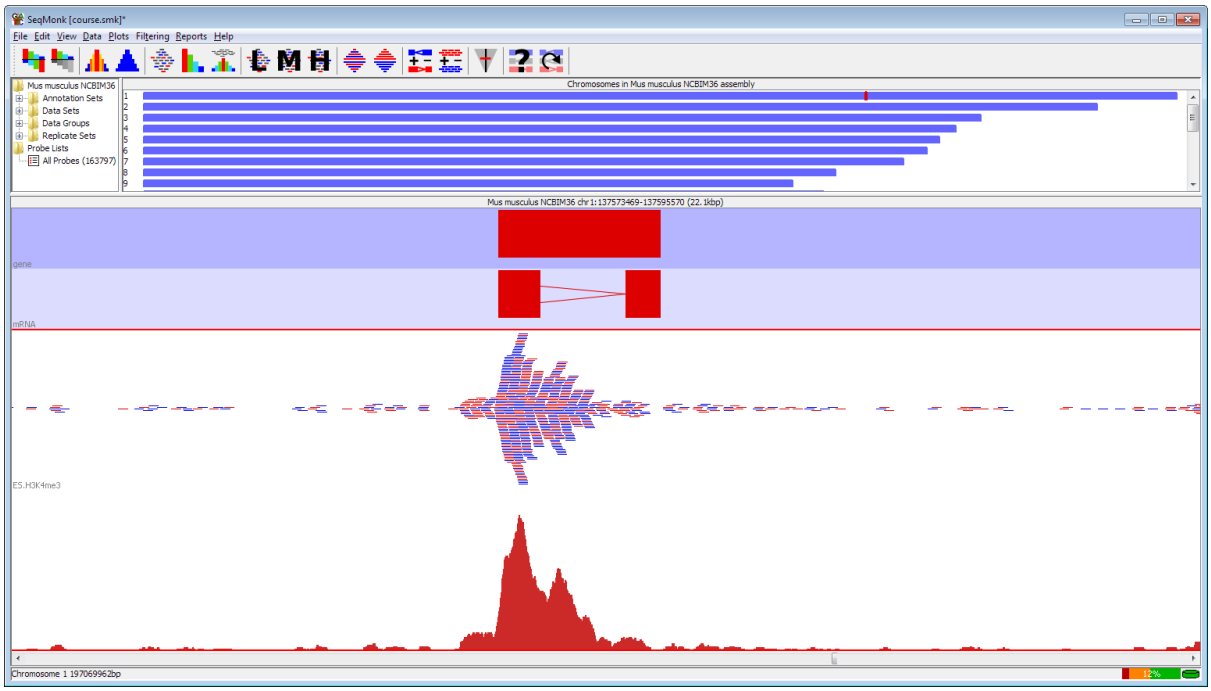
In newer versions of SeqMonk it is possible to construct this kind of view for multiple files to present a high scale overview of your data. The basic process you would use would be:

1. Create running window probes over your genome (or region of interest)
2. Quantitate your data by either:
    a. A read count quantitation (corrected for total count), but not log transformed
    b. An enrichment quantitation
3. Use the fixed colouring view of the chromosome view to make your tracks clearly different

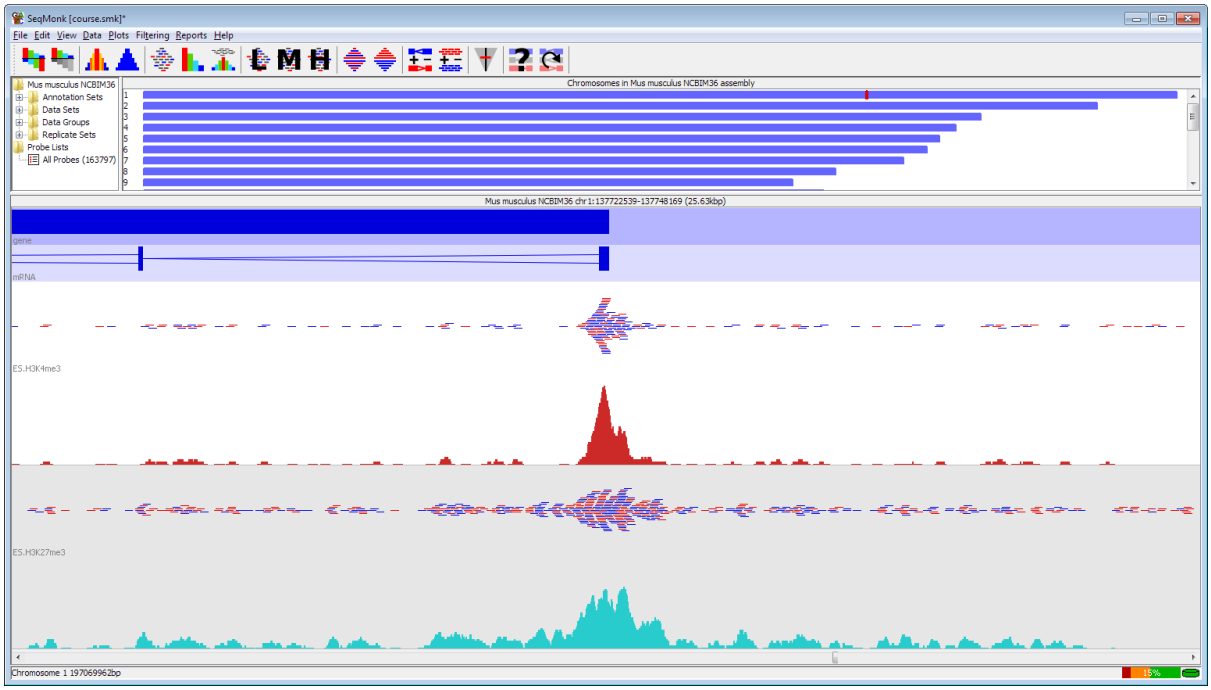
For your running window probes you can choose to either place these over the whole genome, or to just place them within regions of interest. For the whole of a large genome you can generally go down to 200bp windows on most machines. If you have a machine with a reasonable amount of memory then you can probably go down as far as 100bp. If you're looking at large regions of the genome in your figures then you don't need to make the probes too small, and indeed if you make them too small you might lose some of the contrast in your view.



If, on the other hand, you want to look at some fairly targeted regions in more detail with this kind of view then one of the options in the running window generator is to design probes only within the current active probe list. What this means is that you can use any of the other generator options to make a set of candidate regions and then even select a subset of these, and then design smaller running probes only within these regions. Generally it's not a good idea to make your probes too small since you want to avoid having noisy plots. For this type of analysis you could produce very high resolution quantitated maps, even down to single base resolution for small regions (CpG islands, promoters etc.). You can use this in combination with the 'Current Region Probe Generator' to allow you to make high resolution maps of just the region of the genome you're currently looking at. At the bottom end we've found that 5bp probes placed every 1bp are about as small as it's worth going.



When quantitating your data for these plots you can either choose to quantitate absolutely or use an enrichment plot. For absolute quantitation you probably want to use the base pair quantitation to allow for the frequent cases where reads partially overlap your probes. Normally these plots are best viewed on a linear scale to emphasize the differences in your data.



If you want to present enrichment then you should try to ensure that the majority of your probes contain some data, otherwise your empty probes are going to dominate and make the plot look odd. You can use the normal enrichment quantitation to allow an easy view of your overall enrichment, but since this is based on a $\log_2$ scale it can under-emphasise your changes. If you want to put your enrichment on a linear scale then you can do a base pair quantitation, then do a datastore summary report to find the mean quantitation for each dataset. You can then do a manual correction and subtract the mean quantitation from every dataset to get a plot which shows the mean at zero, so enriched regions show positive values, and depleted regions are negative.

Some of the common options for this type of plot have now been made into a pre-built quantitation pipeline which will automatically put a sensible number of probes over your region of interest and do an appropriate quantitation. There's nothing in the pipeline which you couldn't do more manually using the normal quantitation tools, but it makes them more easily accessible.

Gene Reporting and Deduplication

A very common scenario is that you do some analysis in SeqMonk and generate a list of target regions, but at the end of the day what you want to export is a list of genes which you can take forward for functional analysis, looking for pathways, interactions or other biological information to make sense of the hits you saw. A couple of problems commonly rear their head in this scenario.

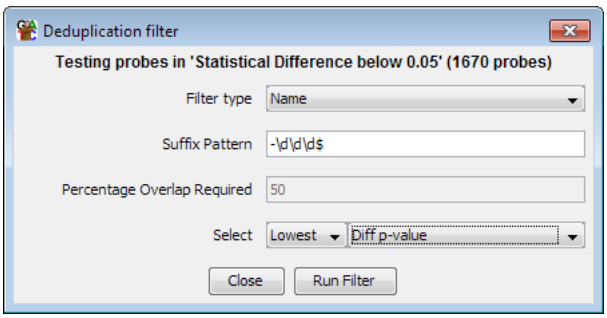
Deduplication

Many analyses will have been performed at the level of the transcript. When seeing hits from these it's common to get multiple hits for the same gene where several splice forms of the same transcript

are found. This can give a misleading impression of the number of true hits in your data, and also produces duplicate output for your subsequent functional analysis.

One way around this is to use the deduplication filtering tool to try to reduce your hits to a single instance of each gene. This filter can remove duplication in one of two ways, either by comparing the names of the probes and looking for commonalities, or by deduplicating based on overlapping chromosomal positions.

The name based deduplication is set up for deduplication of transcripts where the transcripts are named in the form [gene name]-[number]. So for example you might get abc1-002 and abc1-003 being two splice forms of abc1. The named based deduplication allows you to specify a pattern to remove from the probe name, and then any probes which are left with the same name after the removal are assumed to be duplicates. The pattern uses regular expression syntax (which isn't worth going into here, but a quick web search should point you in the right direction), but the default pattern (-\d\d\d$) removes a hyphen followed by 3 digits at the end of the name.



Positional deduplication works by looking for overlaps between probes. Where two probes overlap you can set a threshold of what percentage of either probe must be contained in the overlap region for them to count as duplicates.

Where duplicates occur you can choose how to select the one to keep. You can either base this on the length of the probes so you keep the longest or shortest in each case, or you can use the annotated value on the list you're filtering, so that from a statistical filter you kept the most significant variant for example. Deduplicating in this way will not change the way your probes are reported, or the names which are seen, but it will limit your reports to only showing one transcript per gene rather than listing all of the isoforms individually.

Per-gene reporting

Another, more generic deduplication method is the use of the per-feature report tool. This tool allows you to select what type of feature you want to use for your report, and can then internally combine the results of multiple probes covering that feature to report a single averaged value.

The feature report bases its averaging solely on overlaps between probes and the selected features. If a probe overlaps a feature it can be included in the report. Every probe assigned to a feature will get equal weight in determining the quantitated value reported for that feature. This means that the report is simple to set up, but can be confused by having unrelated, but overlapping probes within a feature (for example sense and antisense transcripts to the same gene).

For a more specific version of the report you can choose to require exactly overlapping probes with the feature. A probe is considered to be exactly overlapping if it's start, end and strand values match the feature being annotated, or one of it's sub-locations in the case of split-location features, exactly.

If you want to have a report on probes which were generated from genes or transcripts, and you want one entry per probe, but you also want to see the correct gene annotated to it then you can use the normal annotated probe report, but when deciding which feature to use you can select 'Name

Matched' as the matching parameter. This will pair probes to features solely based on their name (removing additions such as _upstream or the -001 transcript extensions).