RNAither, an automated pipeline for the statistical analysis of high-throughput RNAi screens

Nora Rieber

December 6, 2009

Contents

1 Introduction

RNAither performs analyses of human RNAi knock-down screens, from raw signal intensities to lists of significant genes and biological processes. There are no prerequisites about plate size, number of signal channels, or availability of control measurements.

An overview of the typical work-flow of the pipeline is given in fig. 1. All steps can be carried out and adapted independently by the user for maximum flexibility. However, to facilitate usage of the pipeline and speed up the analysis, we recommend the use of our wrapper function that performs a comprehensive analysis and presents the results as a set of HTML pages while still allowing to choose the analysis options - for example normalization methods or statistical tests - that are best suited for the type of data at hand in a concrete case.

Section 2 (Generating an input dataset file for RNAither) and 8 (`main-Analysis` wrapper function and HTML output) are relevant for any user wanting to use the automated version of the analysis. Sections 3 through 7 describe the detail of the functions used by the `mainAnalysis` wrapper function, and introduce some additional analysis functions not included in the wrapper, but are irrelevant for the standard use of the package. Section 9 presents a sample application on a genome-wide RNAi screen.

2 Generating an input dataset file for RNAither

The input dataset for the pipeline is a text file generated from the experimental output data that contains all the information necessary to describe the experiment. The text file consists of a header and a table. The header gives information about the external experiment name (`external_experiment_name`), e.g. ""Johns Experiment Nb. 1"", and the type of data (`type_of_data`), e.g. "364 well plate data for virus screens", and allows a space for comments, if any (otherwise `NA`). In the standard case,
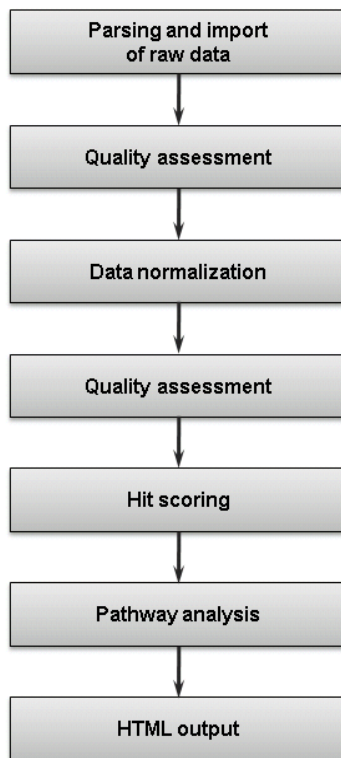
Figure 1: The typical pipeline analysis work-flow. All steps are independent but the full work-flow from quality analysis to html output may be carried out by the `mainAnalysis` wrapper function. HTML output is dependent on the wrapper function.

the table contains 13 columns, each line corresponding to one spot or well on one plate:

- the spot number on the plate `Spotnumber`

- the spot type `SpotType`, which can be 0 (negative control), 1 (positive control), 2 (normal sample), or -1 (empty spot or spot of poor quality that will not be included in the further analysis)

- an internal gene ID (`Internal_GeneID`), for example the siRNA name. Can be equal to `GeneName`.

- the gene name (`GeneName`)

- the signal intensity (`SigIntensity`)

- the standard deviation of the signal intensity (`SDSIntensity`)

- the background intensity value (`Background`)

- the plate number of the spot/well (`LabtekNb`)

- the row number of the spot/well (`RowNb`)

- the column number of the spot/well (`ColNb`)

- the experiment number (`ScreenNb`)

- the number of cells in the spot/well (`NbCells`) - can also be a second intensity channel.

- the proportion of cells among the recognized objects (`PercCells`) - useful when the experimental data is recorded automatically.

These columns are always present, but can be simply set to `NA` if the information is not available. Appending additional columns is unproblematic, and further dataset processing and analysis is not confined to specific column names. During data analysis, results (e.g. normalized values, p-values or hit vectors) are appended as extra columns to the dataset.

We implemented the possibility to make a distinction between `GeneName` and `Internal_GeneID`. All pipeline functions that perform annotations or group signal values according to replicates leave the choice of a classification of replicates according to either the gene (`GeneName`) or the siRNA (`Internal_GeneID`). This means that in case different siRNAs are used to silence

4

the same gene, we can either treat all siRNAs as having the same effect on that gene (i.e. we consider two spots with, respectively, geneA silenced by siRNA1 and geneA silenced by siRNA2, as replicates), or treat them differently (i.e. we only consider two spots with a silenced geneA as replicates if siRNA1 was used for both; geneA silenced by siRNA1 and geneA silenced by siRNA2 are not considered as replicates in the analysis).

Dataset files are usually generated with the function `generateDatasetFile` as illustrated in the following example:

```
> library("RNAither")

> plateLayout1 <- c("test1", "empty", "test3", "test4", "test5",
+     "test6", "test7", "empty", "test9", "test10", "test11", "test12")
> plateLayout2 <- c("test1", "test2", "test3", "test4", "test5",
+     "test6", "test7", "test8", "test9", "test10", "test11", "test12")
> plateLayout <- cbind(plateLayout1, plateLayout2)
> emptyWells <- list(c(2, 8), NA_integer_)
> poorWells <- NA_integer_
> controlCoordsOutput <- list(list(NA_integer_, NA_integer_), list(NA_integer_,
+     c(9, 10)))
> backgroundValOutput <- NA_integer_
> sigPlate1 <- c(2578, NA_integer_, 3784, 3784, 2578, 5555, 5555,
+     NA_integer_, 8154, 2578, 3784, 2578)
> sigPlate2 <- c(8154, 3784, 5555, 3784, 11969, 2578, 1196, 5555,
+     17568, 2578, 5555, 2578)
> meanSignalOutput <- list(sigPlate1, sigPlate2)
> SDmeansignal <- NA_integer_
> objnumOutput <- NA_integer_
> cellnumOutput <- NA_integer_
> generateDatasetFile("First test screen", "RNAi in virus-infected cells",
+     NA_character_, "testscreen_output.txt", plateLayout, plateLayout,
+     3, 4, 1, emptyWells, poorWells, controlCoordsOutput, backgroundValOutput,
+     meanSignalOutput, SDmeansignal, objnumOutput, cellnumOutput)
> header <- readLines("testscreen_output.txt", 3)
> dataset <- read.table("testscreen_output.txt", skip = 3, colClasses = c(NA,
+     NA, NA, NA, "factor", NA, NA, NA, NA, NA, NA, NA, NA, NA),
+     stringsAsFactors = FALSE)
```

Datasets or dataset files from different experiments and/or plates can be joined with the functions `joinDatasets` or `joinDatasetFiles` for a com-

bined analysis:

```
> data(exampleDataset, package = "RNAither")
> doubledataset <- joinDatasets(list(dataset, dataset))
```

    or:

```
> data(exampleHeader, package = "RNAither")
> data(exampleDataset, package = "RNAither")
> saveDataset(header, dataset, "save_testfile1.txt")
> header[[1]] <- "external_experiment_name,Test screen"
> header[[2]] <- "comments,contains twice Screen Nb. 1"
> joinDatasetFiles(list("save_testfile1.txt", "save_testfile1.txt"),
+     3, header, "concatenated_testfile.txt")
```

3 General functions

The pipeline provides a set of functions for handling and restructuring the dataset and the contained values, serving the purpose of adapting the input data to user needs, if required. It can be inserted as an additional pre-processing step between the automated parsing of the experimental data and the data analysis, or be used after the analysis on a fully scored dataset containing normalized values, p-values and hits.

3.1 Creating subsets

The function `createSubset` returns a subset of the given dataset containing all spots/lines of the main dataset that have a certain value in a specified column. For example, it allows to create subsets comprising only spots from a certain plate or experiment, spots containing a specific siRNA or positive/negative controls. The corresponding function `indexSubset` returns the indexes of the concerned lines in the argument dataset.

3.2 Rearranging the dataset

The function `orderGeneIDs` orders the given dataset according to a specified column (usually the signal intensity). `eraseDataSetColumn` deletes the specified column.

3.3 Summarizing intensity values

There are many different ways of summarizing intensity values, e.g. those for one replicate. Besides the obvious `mean` and `median` functions already implemented in R, the pipeline offers the root mean square function (`rms`):

$$RMS = \sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}$$

(x_i, being the intensity values, n their number), as well as a trimmed mean function (`trim`) based on the R function that computes the mean after removing the upper and lower 5% of the values, and the functions `closestToZero` and `furthestFromZero` returning, repectively, the value closest or furthest from zero. These functions can be set as arguments of the function `summarizeReps` that returns a dataset with summarized values (either per siRNA or per gene).

The function `sumChannels` allows to generate an additional dataset column summarizing two columns of the dataset with a specified function (e.g. `divideChannels`). For example, it is possible to divide the column containing the intensity values by the column containing the number of recognized cells.

3.4 Group replicate values

The function `generateReplicateMat`, with the help of the function `findReplicates`, generates a matrix containing either all values of a specific column or all indexes in the dataset for either each possible gene or each possible siRNA. Genes or siRNAs occurring only once in the dataset can be excluded from the matrix, if required. By default, wells with a SpotType set to -1 are ignored. However, it is possible to include those wells, as it may be desirable under certain circumstances.

4 Quality control

The pipeline offers a wide range of possibilities to control the quality of the experiments to analyze. The functions can be used before or after normalization, to assess the quality of raw data or to monitor the effects of normalization.

An easy way of verifying if an experiment was successful is to compare

positive and negative controls. A straightforward way to do so is to compare control intensities with each other and with the remaining experimental data by eye. For this purpose, the functions `makeBoxplotControls` which generates a boxplot with boxes for positive controls, negative controls and remaining data, and `plotControlHisto` that shows a histogram of the data with highlighted controls have been implemented. (If controls are missing, `plotHisto` is used to plot a histogram without highlighting.) All plots are available on screen, experiment and plate level. A numerical measure of the separation of positive and negative controls is given by the Z' factor which is computed by the function `ZPRIMEQualControl` and defined as follows:

$$Z' = 1 - 3\frac{\sigma_{pos} + \sigma_{neg}}{\mid \mu_{pos} - \mu_{neg} \mid}$$

$Z' = 1$ stands for an experiment with optimal separation of controls, $1 > Z' \geq 0.5$ for an experiment with excellent separation of controls, and $Z' < 0.5$ for an experiment with limited quantitative information. The results are plotted and can also be found in an output table. The separation of control densities (on screen, experiment and plate level) can be checked with the plots generated with the function `controlDensity`.

However, positive and negative controls are not always available, or not available on the same plate, in which case experiment quality can also be assessed e.g. by generating a plate plot showing the color-coded spatial distribution of intensity values on the plates with the function `spatialDistrib`. Obvious errors having occurred during the experiment become apparent, for example if certain areas (e.g. the edges of the plate) have strikingly high or low intensity values in comparison with the rest of the plate. `spatialDistrib` uses the plotting function included in the Bioconductor package `prada`. The spots are annotated with HTML mouseovers (using either the siRNA or the gene name) using the Bioconductor package `geneplotter` and controls spots, if available, are marked.

For further processing of the data, it may be important - e.g. if the significance of intensity values is to be assessed by the t-test - to know if it follows a normal distribution. This can be tested by generating a QQ-plot with the function `plotQQ`. Detailed plots per experiment and per plate are available.

Boxplots generated by the functions `makeBoxplotPerScreen`, `makeBoxplotPerPlate` and `makeBoxplot4PlateType` allow a comparison of intensity chan-

8

nels between experiments or plates to evaluate reproducibility, while `plot-Bar` or `ZScorePlotTwo` shows the signal intensity of each well accompanied with the (screen, experiment or plate) median and one and two median absolute deviations. A direct comparison of channels with scatterplots (`channelPlot`) including a lowess regression curve is also possible.

After normalization, the variability of siRNA replicates in each experiment is compared directly through scatterplots (e.g. replicate 1 versus replicate 2) with the function `compareReplicates`, but also with Spearman's rank correlation coefficient, both between experiments and between siRNA replicates (`replicatesSpearmancor`). (A similar function not included in the `mainAnalysis` wrapper function, `compareReplicaPlates`, also allows to compare replicate plates pairwise.) As stated before, "replicates" can be defined by the user either on the gene or the siRNA level.

A further way to assess the reproducibility of siRNA replicates is a measure borrowed from microarray analysis: the coefficient of variation (CV) of an siRNA is defined as the standard deviation of its values divided by their mean:

$$CV = \frac{\sigma_{siRNA}}{\mu_{siRNA}}$$

The function `replicatesCV` plots the CV coefficient versus the mean intensity for each experiment.

Besides, the standard deviation of each siRNA replicate is plotted in the same fashion as the plate plots generated by `spatialDistrib` (see above) with the function `compareReplicateSD`. This way, siRNA replicates and experiments bearing a disproportionate standard deviation are visible at a glance. Plots are available on screen and experiment level.

Additional quality control functions are available that are not included in the `mainAnalysis` wrapper function. For screens conducted with positive and negative controls on each plate, the dynamic range (called with the function `DRQualControl`) is another measure to evaluate the separation of controls:

$$DR = \frac{\mu_{neg}}{\mu_{pos}}$$

μ being the mean of the positive and negative controls, respectively. If there

is a background intensity available, there is also the possibility to compute the signal-to-noise ratio (SNR) with the function `SNRQualControl` which will plot the distribution of SNR's per spot for the complete dataset and for each experiment and plate individually.

If the data was read out with the help of an image recognition software on a cell-by-cell basis, the function `numCellQualControl` allows to set upper and lower thresholds on the number of cells, as an intensity value per spot computed with too little or too many cells might not be meaningful. Accordingly, `percCellQualControl` allows to set upper and lower thresholds on the percentage of objects identified as cells, because if very few objects were identified as cells, the intensity value computed is not meaningful either. The default threshold is 3 standard deviations from the mean and is shown in a histogram. The spot type of spots under or over the respective thresholds is set to -1, discarding them from the further analysis but still keeping them in the dataset file. Additionally, excluded siRNAs are saved to a separate text file.

Wells or entire plates can also be discarded by hand by the user with the functions `discardWells` or `discardLabtek` if there is good cause for doing so, e.g. if experimental evidence suggests the measured values cannot be trusted.

5 Normalization

The dataset column containing the values to be normalized is saved as an additional column with the suffix ".old" (numbered if more than one normalization technique is applied) while the original column is replaced with the normalized values. The name of the methods applied appears in the header comments. As many normalization methods require the assumption that most siRNAs do not have any effect, it is possible to exclude the controls for the computation of normalized values.

5.1 Normalization on controls

The function `controlNorm` allows to normalize values either on the experiment or plate median of either all positive or all negative controls, or to choose a specific control siRNA or gene to normalize values on.

5.2 Normalization on mean, median, etc

The function `divNorm` accepts any kind of summarization function specified in R, e.g. the mean or median, and applies it either to experiments or to plates. Depending on requirements, the summarized value is subtracted from each value (e.g. in the case of log-values) or divides it.

5.3 Normalization with Z-scores

For each spot i, the Z-score is defined as:

$$Z = \frac{x_i - \overline{x}}{s_x}$$

x_i being the signal intensity for spot i, $\overline{x}$ the mean intensity value of the plate and s_x the standard deviation. For the function `ZScore`, these were replaced by the more robust alternatives of median and median absolute deviation. Also, a Z-score per screen (instead of per plate) is available.

5.4 Normalization with B-scores

The B-score is implemented in the pipeline with the function `BScore`. The procedure first calculates the so-called residual r_{ijp} for row i and column j on plate p:

$$r_{ijp} = y_{ijp} - \hat{y}_{ijp} = y_{ijp} - (\hat{\mu}_p + \hat{R}_{ip} + \hat{C}_{jp})$$

y_{ijp} being the measured value, $\hat{y}_{ijp}$ the value fitted by a two-way median polish that estimates systematic measurement offsets for each row i ($\hat{R}_{ip}$ being the median of row i) and column j ($\hat{C}_{jp}$ being the median of column j). $\hat{\mu}_p$ is the estimated average of the plate.

The B-score is then calculated with:

$$BScore = \frac{r_{ijp}}{MAD_p}$$

MAD being the median absolute deviation defined as follows:

$$MAD_p = median\{|r_{ijp} - median(r_{ijp})|\}$$

5.5 Quantile normalization

Quantile normalization is implemented in the pipeline with the function `quantileNormalization`, which uses the Bioconductor package `limma`.

Given n plates or experiments with p values, a matrix X with p rows and n columns is formed. Then each column of the matrix is sorted according to its values and each element of a row is replaced with the mean of the row. Finally, the elements of each column are sorted back into their initial order.

5.6 Li-Wong rank normalization

The Li-Wong rank normalization - also called invariant probe set normalization - has its origin in microarray pre-processing and is implemented in the pipeline with the function `LiWongRank`. The idea is that, given a ranked list of signal intensities, spots having the same or nearly the same rank on every replicate plate form the "invariant probe set" and are well-suited for normalization.

Our pipeline uses a modified version of the technique since the original method was designed to deal with two-color microarrays, i.e. two linked data channels, which is not the case in siRNA experiments.

Our function is designed to normalize experiments, each with a certain number of plates n, that have been repeated several times. For each plate type in each experiment, the siRNAs are sorted according to their value on the plate. Only siRNAs used once on the plate are taken into account. (Again, it is possible to differ between siRNAs and genes.) It is obvious that plate designs where each siRNA is used several times are not suited for an invariant probe set normalization. An error is triggered if the plate layouts are not the same for each experiment, or if more than 20% of the spots on one plate are occupied by siRNAs occuring several times on the plate.

Subsequently, for each siRNA the standard deviation of its ranks is computed in order to select siRNAs with a very low standard deviation of ranks, i.e. having approximately the same rank on each replicate plate, to form the "invariant probe set".

The function prints out a histogram of the standard deviations of ranks for each plate type and allows the user to select one out of a series of siR-

NAs with low standard deviations. For each plate, each spot value is divided by the value of the spot containing the chosen siRNA.

5.7 Lowess normalization

The Lowess normalization, or locally weighted polynomial regression, is used in the specific case of two data channels that are assumed to be independent of each other, e.g. one for a signal intensity, the other for the cell count per spot. Normally, the signal intensity should not be dependent on the cell count, however, plots of one channel versus the other show that sometimes this seems to be the case.

The Lowess normalization is a technique from microarray normalization that fits a smoothing curve through the points. The idea is to down-weigh data points. In our example, the normalization would down-weigh the signal intensity of data points which are more than a certain percentage away from the signal mean. The Lowess normalization is implemented in the pipeline with the function `lowessNorm`.

5.8 Additional data processing

Some post-processing steps, for example certain statistical hypothesis tests, might require an additional normalization of the signal variance. Additionally, it has been shown on microarray data that signal variance increases with the signal. Variance normalization can be done by dividing values by the plate or experiment median absolute deviation with the function `varAdjust`.

The function `subtractBackground` can be used to subtract signal background from each well, if applicable.

The dataset can be saved to a text file with the function `saveDataset`. However, this is not necessary when using the `mainAnalysis` wrapper function.

5.9 Which normalization option to choose?

The first question to ask is whether we expect most siRNAs to have a significant effect on the output signal, which for example is the case in validation screens. If this is the case, most normalization methods cannot be used since they assume the majority of siRNAs on one plate can serve as controls. As an example, the Z-score rescales signal intensities relative to within-plate

variation, which in this case is of course not a desired effect.

If controls are available on each plate, one possibility is to normalize values based on them. However, if only few controls are available, the risk of introducing severe errors is higher than the potential benefit of the control normalization.

In case there are few or no controls available, an option is to use the Li-Wong rank normalization which finds "stable" siRNAs, i.e. siRNAs that are not used as controls but can serve as such.

If we can make the assumption that most siRNAs will not have an effect, more normalization options are available. When the plates can be assumed to be similar (i.e. because they have the same siRNA layout or a high number of different siRNAs in a primary screen), a normalization on the plate mean or median (more robust) is acceptable.

A normalization with Z-scores introduces an additional variance normalization and can allow to score hits without using statistical hypothesis tests, e.g. by stating every siRNA as a hit that is more than two standard deviations (median absolute deviations) away from the population mean (median). This can be useful when only few replicates of each siRNA are available, which would make a statistical hypothesis test unreliable.

On large plates ($\geq$ 96 wells) a within-plate normalization can be necessary as we encounter the problem of position (e.g. row and column) bias, for example through edge effects. This can be counteracted by using a normalization partially similar to the Z-score, the B-score, which estimates row and column biases and corrects them.

The quantile normalization is similar to the Z-score in that it adjusts the distribution of values on each plate/experiment so that they are more or less the same.

Finally, in the case of two independent signal channels, the Lowess normalization can be used to correct a dependency artifact.

6   Statistical tests and hit scoring

Hits can be scored according to the t-test (function `Ttest`), Mann-Whitney test (function `MannWhitney`), and/or the Rank Product test (function `RankProduct`). P-values can be corrected for multiple testing with `multTestAdjust`.

Each test function is applied to all values of each replicate (biological or technical) in a specified channel. A sorted, named vector of p-values can be saved to a text file with the function `savepValVec`. P-values can be added as an extra column to the dataset with `incorporatepValVec`.

Subsequently, genes or siRNAs can be selected as hits with the function `hitselectionPval`, `hitselectionZscore`, or `hitselectionZscorePval`. In the first case, all p-values under a user-defined threshold are chosen. The second case can be applied when the hit selection is performed directly on signal values (either per spot or per gene/siRNA), for example Z-scores, without the use of a statistical test. The user can choose if he wants the first or last n values of a sorted list of signal values (a warning is issued if the length of the list is less than $2n$), or all values above or below a certain threshold. The function `ZScorePlot` helps with the choice of the threshold by plotting the normalized signal values of all spots or replicates together with the mean and one and two standard deviations. The third case allows hit selection according to both a threshold for p-values and one for signal intensities. This can be useful when there are only few replicates available, and allows to exclude hits that are only scored because of the small standard deviation of the signal intensity values of the corresponding replicates.

In all cases, a binary hit vector is generated and added as an extra column to the dataset. A warning is issued if no hits were identified according to the user-defined threshold, which is then changed in order to select at least one hit. A text file containing the identifiers, the corresponding p-values or in the second case the summarized signals, and all individual signals in the screen is generated. Calling the function `spatialDistribHits` generates plate plots of the individual plates, showing spots identified as hits, which allows to identify suspicious distributions of hits on the plate (e.g. all in one column or one area of the plate).

If a statistical test was performed, it is possible to generate a modified volcano plot (with the function `volcanoPlot`), i.e. a plot of the normalized signal values versus the negative decadic logarithm of the corresponding p-

value. A horizontal line is drawn at $-log_{10}$(p-value-threshold) to see hits at a glance.

Finally, when different tests were performed, it is useful to compare the hits scored with different methods, which can be achieved with the functions `vennDiag` and `compareHits`. Venn diagrams show the number of overlaps from up to three different scoring methods while the hit comparison function saves a list of hit genes or siRNAs that were common to two scoring methods to a text file. These functions also make it possible to compare hits from two datasets screened under different conditions, provided the siRNAs used are the same. Hits can be ordered according to their IDs with the function `orderGeneIDs` cited in 3 and then a Venn diagram can be plotted.

6.1 Which options to choose?

The standard way of assessing whether siRNAs have a significant positive or negative effect on the signal is the use of statistical tests. However, when few or no replicates of an siRNA are available, as it can happen in very large, expensive screens, statistical tests are not viable.

In this case, hits can be defined as values that are more than a certain number (typically two) of standard deviations (median absolute deviations) away from the population mean (median). This, of course, is only an option if we can make the assumption that most siRNAs do not show an effect (see 5.9). Also, it requires prior normalization of mean (median) and standard deviation (median absolute deviation), e.g. with Z-score or B-score. This scoring method can also prove useful when combined with a statistical test (see 6).

When enough replicates are available (number of replicates ≥ 5), we can choose from three different statistical tests: the t-test, the Mann-Whitney test, and the Rank Product test. The first is parametric and requires that the values follow the normal distribution, the latter ones are non-parametric.

If we can prove that the data follows a normal distribution (e.g. with a QQ-plot taken from the quality control), the t-test is the method of choice. If we cannot, a Mann-Whitney test is possible, as it does not make assumptions about underlying distributions and is less susceptible to produce wrong results in the presence of outliers.

The Rank Product test is a very intuitive test that has become popular over the last years. It is claimed to be reliable even in highly noisy data, but needs a relatively high number of replicates to be reliable.

7 Gene Set Enrichment Analysis

The main part of the pathway analysis, i.e. searching for overrepresented pathways among the hits, is done with the Bioconductor package `topGO`. The function first annotates each gene name with its corresponding human GO identifiers with the Bioconductor package `biomaRt`.

The actual pathway analysis is done with the function `gseaAnalysis` which calls the functions of the `topGO` package. It uses the hit vector generated as described in 6 and leaves the choice of one of the three GO ontologies - biological process, molecular function and cellular component. `topGO` uses an algorithm that accounts for the hierarchical structure of the GO database by filtering out local dependencies that lead to redundancy.

The analysis results are summarized in a table including the top GO terms identified by the algorithm, their IDs, the corresponding p-value, and the number of genes in the analysis annotated with this GO term compared to the number of significant genes annotated with this GO term.

8 `mainAnalysis` wrapper function and HTML output

While the user is at liberty to use and assemble all available pipeline functions as he sees fit, a wrapper function that implements a typical work flow and presents the analysis results in a set of HTML pages is on hand. The specification of:

- the header and dataset

- one or several normalizations to perform, along with the normalization functions' arguments (e.g. specification of channel, annotation, etc.)

- one or several statistical tests to apply to the normalized data, along with the test functions' arguments (e.g. specification of channel, annotation, etc.)

- a correction for multiple testing (can be "none")

- which type(s) of hit scoring to use (e.g. according to p-value and/or Z-score)

- which threshold(s) to use for hit scoring

- whether controls are available or not

- the names of the signal channels in the dataset file

- the GO hierarchy to choose for the GSEA analysis (`biological_process`, `molecular_function` or `cellular_component`)

allows the `mainAnalysis` function to perform a comprehensive analysis of the data as follows: First, the quality of the raw data is assessed with the functions described in 4. The results are displayed as an overview on the HTML page `index.html`. More detailed plots (i.e. on single experiment or single plate level) are available by clicking on the overview plots.

The `mainAnalysis` wrapper function then carries out all normalizations specified by the user. After each normalization, another quality check analogous to the first one is conducted and siRNA replicates compared (as described in 4) on the HTML pages `indexnorm.html`. Then, statistical hypothesis tests are performed and hits scored as specified by the user. Finally, a gene set enrichment analysis (GSEA) searches for overrepresented biological processes among the hits. Results of the analysis and plots showing the overlap between different testing and scoring methods, if applicable, are shown on the HTML page `stats.html`, as well as a link to the text file containing the fully scored dataset, and a link to the GSEA analysis output table.

The following example analyses a dataset as follows: the signal intensities in the column `SigIntensity` are normalized on the experiment median of the negative controls. Hits are scored with the t-test and the Mann-Whitney test. Both tests are carried out twice, once for significantly low intensity values, once for significantly high intensity values (both compared to the intensity median 1). There is no correction for multiple testing. The p-value threshold chosen is 0.05. We specify two data channels, "SigIntensity" and "NbCells". GSEA analyses are carried out for each scoring method, based on the "biological process" branch of Gene Ontology.

```
data(exampleHeader, package="RNAither")
data(exampleDataset, package="RNAither")
```

The wrapper function should then be called as follows:

```
mainAnalysis(header, dataset, 0, list(controlNorm), list(list(1,
0, "SigIntensity", 1)), list(Ttest, MannWhitney, Ttest, MannWhit-
ney), list(list("l", 1, "SigIntensity", "GeneName"), list("l", 1,
"SigIntensity", "GeneName"), list("g", 1, "SigIntensity", "Gene-
Name"), list("g", 1, "SigIntensity", "GeneName")), "none", c(1,
0, 0), c(0.05, 0, 0), 1, 1, c("SigIntensity", "NbCells"), "bio-
logical_process")
```

9 A sample application on a genome-wide RNAi screen

We chose a dataset of a genome-wide RNAi screen of cell viability in Drosophila cells (M. Boutros et al., Genome-wide RNAi analysis of growth and viability in Drosophila cells, Science, 303(5659):832-835, 2004.). Genes were knocked down and cell viability evaluated by measuring luciferase activity, which was used as a representative of ATP levels in the cells.

The screen consists of 2 times 57 plates (384 wells). Each plate contains one negative and one positive control. The dataset is available in the RNAither package and can be loaded with the commands:

```
data(headerDrosophila, package="RNAither") data(datasetDrosophila,
package="RNAither")
```

This is a primary screen with a small number of controls per plate and a small number of replicates (two for most of the siRNAs). Given that the plates are rather large, the best adapted normalization is the B-score (see 5.9). The most suitable hit scoring method is a scoring according to the normalized signal intensities (see 6.1). We define hits as any siRNA yielding a median replicate value smaller than -2 (i.e., more than twice the median absolute deviation away from the population median). An additional rank product test was carried out for comparison purposes. A GSEA analysis was not conducted since it is not of primary interest in the context of a whole-genome cell viability screen.

The command to conduct the analysis described is the following:

```
mainAnalysis(headerDrosophila, datasetDrosophila, 0, list(BScore),
list(list("SigIntensity", 1)), list(RankProduct), list(list(100,
```
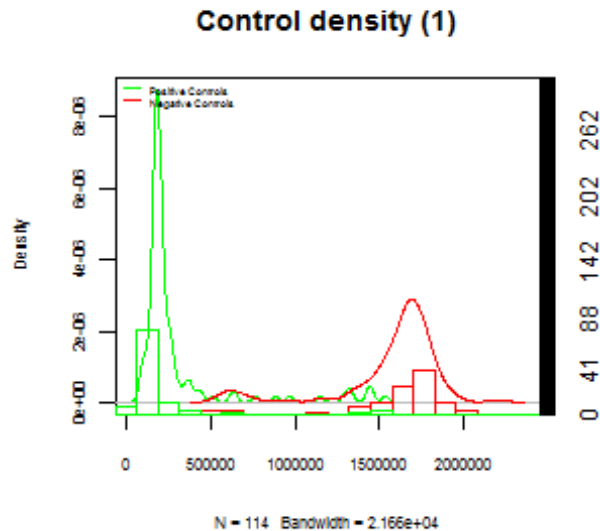
19

Figure 2: The overall control density of the screen before normalization. The superimposed histogram shows the actual proportions.

```
1, "SigIntensity", "Internal_GeneID")), "none", c(1, 1, 0), c(0.05,
-2, 0), 1, 0, "SigIntensity", NA)
```

9.1 Analysis results

We present an excerpt of the analysis covering the most interesting points. Figure 2 shows the overall control density of the screen before normalization. We can see that the separation of positive and negative controls is rather good, which speaks in favour of the good quality of the experiment.

Figure 3 shows a plot of the signal intensities on the first plate of the first experiment before normalization. The rows on the edge of the plate are consistently darker than the rest of the plate, very probably due to edge effects. This effect also appears visibly on other plates. Figure 4 shows a plot of the signal intensities on the first plate of the first experiment after B-score normalization. We can see that the edge effects have been corrected.

Figure 5 shows a qq-plot against the normal distribution of all data before normalization. We can see that the data does not tend to follow a normal distribution, which excludes a t-test for hit scoring altogether.

Figure 6 shows a screen shot of the hits, based on the difference of their
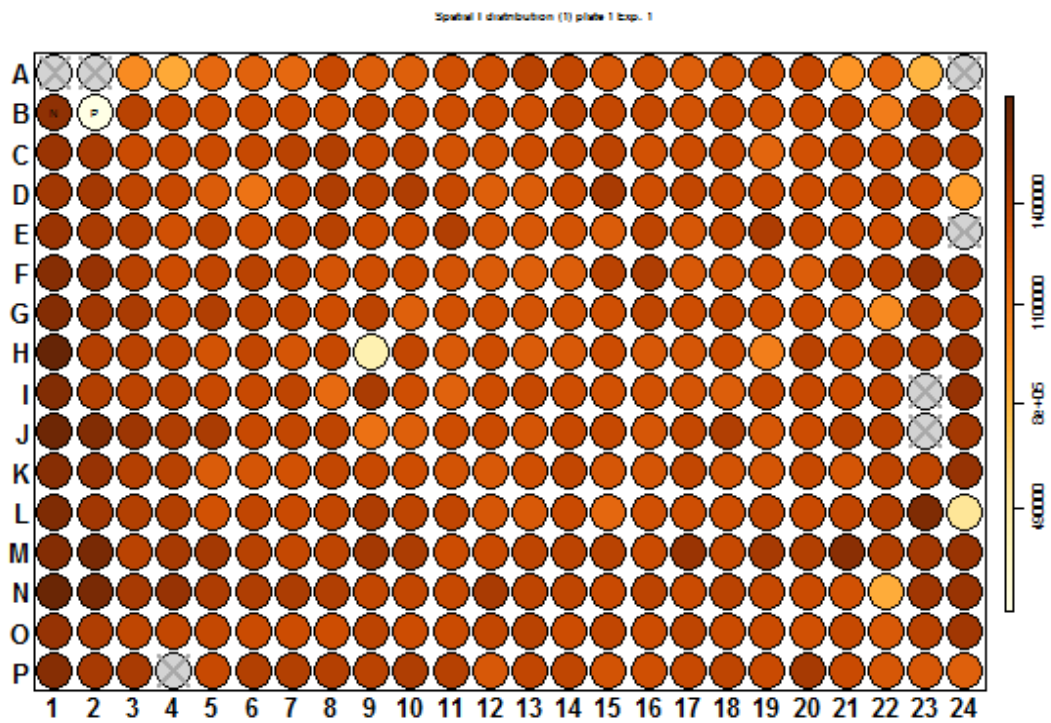
Figure 3: A plot of the signal intensities on the first plate of the first experiment before normalization. "N" and "P" stand for negative and positive controls, respectively, and crossed-out wells mean the signal intensity is not defined (NA).

Figure 4: A plot of the signal intensities on the first plate of the first experiment after B-score normalization. "N" and "P" stand for negative and positive controls, respectively, and crossed-out wells mean the signal intensity is not defined (NA).
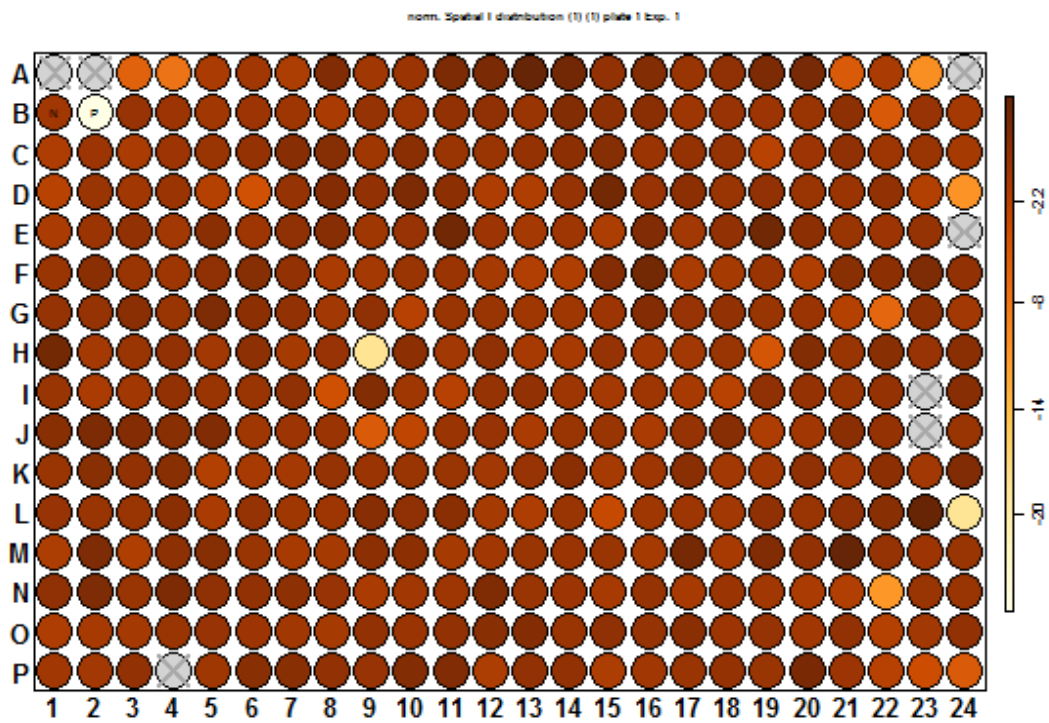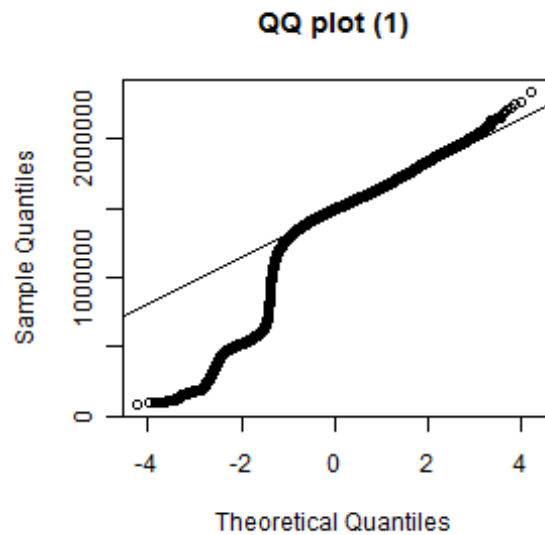
Figure 5: A QQ-plot against the normal distribution of all data before normalization.

median to the screen median, on the html results page. We can see that the positive control is high among the results. Figure 7 shows the hit distribution on the first plate. No suspicious distributions can be found (e.g. an accumulation of hits in one row or certain part of the plate).

Downregulated genes according to ZScore_do

(Threshold: Z-score < -2) - Textfile -

Gene name	ZScore (median)	Norm val	Norm val	Norm val	Norm val	Norr
CG12223	-29.0516386546135	-29.563570654954	-28.5397066542731	NA	NA	N
CG11700	-29.0068537811583	-30.5994356519795	-27.4142719103372	NA	NA	N
CG13235	-28.890898986896	-26.6860772923	-31.095720681492	NA	NA	N
CG12912	-28.1064939517734	-29.4616365650609	-26.7513513384859	NA	NA	N
poscontr	-27.4205512948812	-25.4142391983907	-19.4806017752727	-19.8739406230241	-18.6012186786808	-28.13734
CG33541	-26.3746535923344	-20.9609704755136	-31.7883367091551	NA	NA	N
CG7105	-25.9333800253393	-26.6910901926803	-25.1756698579982	NA	NA	N
CG32970	-25.8554903566564	-23.1245749560331	-28.5864057572796	NA	NA	N
CG32606	-25.0654840777195	-25.4186401306353	-24.7123280248038	NA	NA	N
CG8367	-24.5581572255144	-25.3295022718662	-23.7868121791626	NA	NA	N
CG12284	-24.3344410940106	-27.1878624067427	-21.4810197812786	NA	NA	N
CG15470	-22.9877093150941	-22.9242364324343	-23.0511821977539	NA	NA	N
CG3075	-21.8468742502507	-21.8710833782894	-21.8226651222121	NA	NA	N
CG7552	-21.801780937683	-19.759920414669	-23.8436414606971	NA	NA	N
CG5166	-21.7032937290208	-19.8477762680692	-23.5588111899724	NA	NA	N
CG13165	-21.6211955532516	-19.1519579358233	-24.0904331706799	NA	NA	N
CG6884	-21.4632402184114	-20.8753121078918	-22.051168328931	NA	NA	N
CG31395	-21.3226022071717	-19.7658203096011	-22.8793841047424	NA	NA	N
CG13222	-21.2729124601345	-20.3126504420753	-22.2331744781938	NA	NA	N

Done

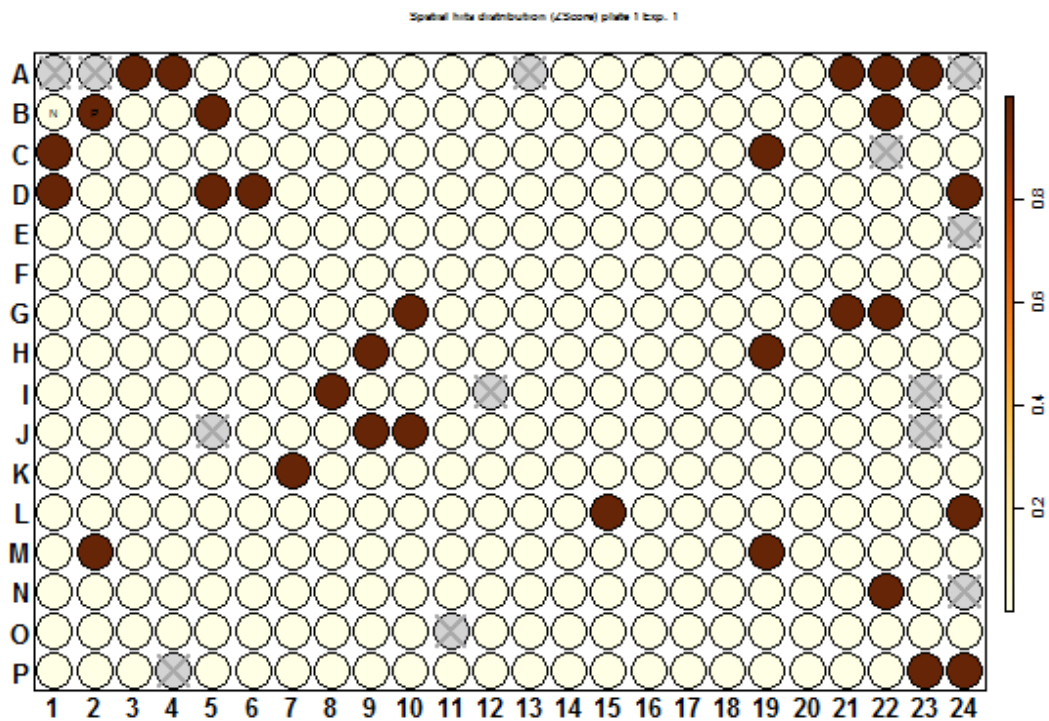Figure 6: a screenshot of the hits on the html results page.

Figure 7: The hit distribution on the first plate. Dark spots symbolize hits. "N" and "P" stand for negative and positive controls, respectively, and crossed-out wells mean the signal intensity or gene name were not defined (NA).