Statistical Integration of Microarrays

Renée X. de Menezes, Marten Boetzer, Melle Sieswerda, Judith M. Boer
Center for Human and Clinical Genetics,
Leiden University Medical Center, The Netherlands
Package SIM, version 1.8.6
R.X.Menezes@LUMC.NL

April 21, 2009

Contents

1 Introduction

This package implements the methods described in [5]. Briefly, we propose the use of a random-effects model to fit the association between copy number and gene expression microarray data, measured on the same samples but not necessarily using the same array platform. The model (and this package) can be applied to either intensity or ratio data. Moreover, it can be used to describe the association between any other two microarray data sources, such as methylation and expression, or SNP call and expression, for example. For simplicity, we will focus on the association between copy number and gene expression.

The *SIM* package depends on the *globaltest*, *marray*, and *quantsmooth* Bioconductor packages, as well as on the R packages *fields*, *spam*, and *clac* from CRAN.

2 Overview

The package consists of functions to run and visualize results of an integrated analysis model being applied to two microarray datasets. Two main functions read in the data and fit the model. Then results can be visualized and tabulated with the help of other functions. We will describe the functions in more detail in the example in section 4. Here we give a brief overview of the functions implemented.

The `assemble.data` function reads in the microarray data and annotation. The main function `integrated.analysis` fits the model to the data. Both functions generate results that are automatically stored on the hard disk for subsequent analysis, in especially created folders. All subsequent functions produce output directly saved onto these folders, so no graphs are displayed using the R graphics window. This is more efficient than keeping large objects on the workspace, or displaying the graphs on the screen, especially if high-density arrays are involved.

The function `sim.plot.pvals.on.genome` gives an overview of the multiple-testing corrected p-values, organized along the genome. Another summary is produced by `tabulate.pvals`, a tabulation of the multiple-testing corrected p-values per studied region. The function `sim.plot.pvals.on.region` generates a multipage pdf including all tested regions, for which it displays the empirical distribution of the computed p-values to convey the strength of the associations found. On the following page, it displays the multiple-testing corrected p-values per region studied, without discretization. This helps identifying regions rich in low p-values. If there is no prior interest in any chromosome, researchers may find it useful to use these graphs, together with the tabulated p-values, to choose chromosome arms to focus on.

The function `sim.plot.zscore.heatmap` plots the pairwise associations between features in the analyzed region in a heatmap. An additional panel can display the dependent features trend on the samples used, in this example the copy number aberrations. Finally, to select dependent or independent features for further analysis and validation, the functions `tabulate.top.dep.features` and `tabulate.top.indep.features` produce tables of dependent features with adjusted p-values and independent features with mean associations, respectively.

3 Data preparation

The microarray data sets should have been normalized with an appropriate method prior to the integrated analysis and contain log-transformed intensities or ratios. In addition to the columns containing normalized microarray measurements, the minimal probe annotation required is a unique identifier, chromosome number (convert X to 23 and Y to 24), and base pair location within the chromosome. Optionally, an additional identifier, often gene symbol, can be provided.

Make sure that the genomic locations in both datasets refer to the same genome assembly and that this is in accordance with the genome build used in the chromosome table used by the *SIM*

package. Currently, the `chrom.table` is available for homo_sapiens_core_40_36b. The function `sim.update.chrom.table` gives instructions on how to change the chrom.table used.

Often, the annotation for expression array probes lack chromosome position information. We generated two methods to add this annotation. The `link.metadata` function gets annotation out of a AnnotationData package and links it to the expression data using the expression probe IDs. It adds the two required columns chromosome and position to run the `integrated.analysis`. An optional column, "Symbol" can be added. Alternatively, we can use the *SIM* function `RESOURCERER.annotation.to.ID`, which gets annotation out of a RESOURCERER annotation file and links them to expression data with help of expression IDs.

We recommend applying the model on the normalized (typically logarithmic) data directly, since the use of discretization may dampen small associations. However, if for some reason the dependent data has to be categorized, it is advisable to transform it into a factor so that the model will use the appropriate settings (see subsection 5.1).

The function `assemble.data` currently expects data frames as inputs representing the array datasets. If your array data is stored as an `exprSet` object, you may use `exprs` to extract the array intensities/log-ratios only. If your array data is just a tab-delimited text file, reading it using `read.table` will produce a data frame.

The order of the samples within each dataset is assumed to be the same and the column names should be identical. For an example how to re-order the samples see 4.1.

The dependent data should not contain any `NA`s. We have constructed the function `impute.nas.by.surrounding` to impute missing copy number data by taking the median of the surrounding probes within the same sample. This function may take a while!

4 Example: Breast cancer

4.1 Data sets

As an example we use the data generated and first analysed by Pollack and colleagues [6]. This example involves array CGH and expression arrays from 37 breast tumor samples and 4 breast cancer cell lines. Just for illustrative purposes we use only chromosome arm 8q. This study used the same cDNA arrays for both expression and CGH measurements, resulting in a one-to-one correspondence between the datasets. This is not a requirement to run the integrated analysis, but it makes biological interpretation somewhat easier.

Arrays from each of the two data types have been pre-processed as follows. A sliding window (size = 5) was applied to the array CGH data. If an `NA` was found at the center of the window and if it was the only `NA` in the window, it was imputed by taking the median of the remaining 4 features. If the window contained more than 1 `NA`, the feature was skipped. After `NA` imputation, all rows (aCGH features) containing `NA`s were discarded. The expression data was filtered to make the expression features correspond to the aCGH features (i.e. expression features that were no longer in the aCGH dataset after `NA` filtering, were discarded). The end result is two datasets with an equal number of rows. The data had been previously normalized by the authors of the study.

The Pollack data is available from the *SIM* package. An overview of the variables and data contained in the package can be obtained via the command

```
> help(package = "SIM")
```

To load the package after its installation, use

```
> library(SIM)
```

Then the expression and copy number data can be uploaded via the commands

```
> data(expr.data)
> data(acgh.data)
> data(samples)
```

The objects `expr.data` and `acgh.data` are data frames, and they include the probe annotations. To find out which columns they contain, use

```
> names(expr.data)
```

```
 [1] "ID"           "Symbol"       "CHROMOSOME"   "STARTPOS"     "Abs.start"
 [6] "BT474"        "MCF7"         "NORWAY.10"    "NORWAY.100"   "NORWAY.101"
[11] "NORWAY.102"   "NORWAY.104"   "NORWAY.109"   "NORWAY.11"    "NORWAY.111"
[16] "NORWAY.112"   "NORWAY.12"    "NORWAY.14"    "NORWAY.15"    "NORWAY.16"
[21] "NORWAY.17"    "NORWAY.18"    "NORWAY.19"    "NORWAY.26"    "NORWAY.27"
[26] "NORWAY.39"    "NORWAY.41"    "NORWAY.47"    "NORWAY.48"    "NORWAY.53"
[31] "NORWAY.56"    "NORWAY.57"    "NORWAY.61"    "NORWAY.65"    "NORWAY.7"
[36] "SKBR3"        "STANFORD.14"  "STANFORD.16"  "STANFORD.17"  "STANFORD.2"
[41] "STANFORD.23"  "STANFORD.24"  "STANFORD.35"  "STANFORD.38"  "STANFORD.A"
[46] "T47D"
```

```
> names(acgh.data)
```

```
 [1] "ID"           "Symbol"       "CHROMOSOME"   "STARTPOS"     "Abs.start"
 [6] "BT474"        "MCF7"         "NORWAY.10"    "NORWAY.100"   "NORWAY.101"
[11] "NORWAY.102"   "NORWAY.104"   "NORWAY.109"   "NORWAY.11"    "NORWAY.111"
[16] "NORWAY.112"   "NORWAY.12"    "NORWAY.14"    "NORWAY.15"    "NORWAY.16"
[21] "NORWAY.17"    "NORWAY.18"    "NORWAY.19"    "NORWAY.26"    "NORWAY.27"
[26] "NORWAY.39"    "NORWAY.41"    "NORWAY.47"    "NORWAY.48"    "NORWAY.53"
[31] "NORWAY.56"    "NORWAY.57"    "NORWAY.61"    "NORWAY.65"    "NORWAY.7"
[36] "SKBR3"        "STANFORD.14"  "STANFORD.16"  "STANFORD.17"  "STANFORD.2"
[41] "STANFORD.23"  "STANFORD.24"  "STANFORD.35"  "STANFORD.38"  "STANFORD.A"
[46] "T47D"
```

The sample columns should have identical names in both data sets, also they should be ordered the same way. To order them, use the `order` function on a data frame containing the sample data only. After sorting, the annotation columns are added again.

```
> acgh.data.only <- acgh.data[, 5:ncol(acgh.data)]
> expr.data.only <- expr.data[, 5:ncol(expr.data)]
> acgh.data.s <- acgh.data.only[, order(colnames(acgh.data.only))]
> expr.data.s <- expr.data.only[, order(colnames(expr.data.only))]
> sum(colnames(expr.data.s) == colnames(acgh.data.s))
```

```
[1] 42
```

```
> acgh.data <- cbind(acgh.data[, 1:4], acgh.data.s)
> expr.data <- cbind(expr.data[, 1:4], expr.data.s)
```

4.2 Assembling the data

The `assemble.data` function helps you read in the measurements and annotation data and stores them for use in the integrated analysis and visualization. Also, the `assemble.data` function takes care of ordering the probes according to position along the genome by giving each probe a unique location called absolute start, generated by chr*10e9+basepair.

Finally, in `assemble.data` you define the `run.name`; a folder with this name will be generated in which subfolders will hold the data and output. In this example, we will do the integrated analysis for chromosome 8q, as the `run.name` indicates.

```
> assemble.data(dep.data = acgh.data, indep.data = expr.data, ann.dep = colnames(acgh.data)[1:4],
      ann.indep = colnames(expr.data)[1:4], dep.id = "ID", dep.chr = "CHROMOSOME",
      dep.pos = "STARTPOS", dep.symb = "Symbol", indep.id = "ID",
      indep.chr = "CHROMOSOME", indep.pos = "STARTPOS", indep.symb = "Symbol",
      overwrite = TRUE, run.name = "chr8q")

[1] "the inserted chromosome column of the independent dataset is numeric"
[1] "the inserted start position column of the independent dataset is numeric"
[1] "Trying to generate an absolute start object"
[1] "Succesfully generated an absolute start object"

[1] "the inserted chromosome column of the dependent dataset is numeric"
[1] "the inserted start position column of the dependent dataset is numeric"
[1] "Trying to generate an absolute start object"
[1] "Succesfully generated an absolute start object"
```

4.3 Applying the model

In this example, the main objective is to identify candidate regions whose copy number aberrations affect expression levels of genes in the same region. Therefore, it is natural to consider copy number as the dependent variable per cDNA clone, with the expression levels of genes in the same region, playing the role of independent variables.

The integrated analysis is a regression of the independent data on the dependent features. The regression itself is done using the `globaltest` [4], which means that the genes in a region (e.g. a chromosome arm) are tested as a gene set. The individual associations between each copy number probe and each expression probe are calculated as z-scores (standardized influences, see `?globaltest`).

The global test can calculate the p-values using different models, the most important ones being the `asymptotic distribution` (recommended for large sample sizes; can be conservative for small sample sizes) and `permutations` (recommended for smaller sample sizes or when the asymptotic distribution cannot be assumed). The `gamma` method can also be used, but as it tends to be anti-conservative, we advise against it. The method to be used can be specified using the option `method`. The default `auto` uses the exact permutation method if the number of possible permutations is less than 10,000 and the `asymptotic` method otherwise. When confounders are included in the model, it is not possible to use the `permutation` method, so the `asymptotic` method is suggested.

The user is free to choose the regions of interest. For an unbiased, genome-wide view, we recommend using chromosome arms. Predefined input regions are `"all arms"` and `"all arms auto"` for autosomal chromosome arms only. The arms 13p, 14p, 15p, 21p and 22p are left out, because in most studies there are no or few probes in these regions. To include them, just make your own vector of arms. Similarly, `"all chrs"` and `"all chrs auto"` may be used. When minimal common regions of gains and losses have been defined, the integrated analysis can be focussed to identify candidate genes in these regions, defined by the chromosome number followed by the start and end position like `"chr1_1-1000000"`. These regions can also be combined, e.g. `c("chr1_1-1000000","2q", 3)`. The function splits the datasets into separate sets for each region (as specified by the `input.regions`) and runs the analysis for each region separately.

When running *SIM* for a predefined input region, like `"all arms"`, output can be obtained for all input regions, as well as subsets of them. But note that the genomic unit must be the same: if `integrated.analysis` was run using chromosome arms as units, any of the functions and plots must

also use chromosome arms as units, and not e.g. chromosomes. For example if the `input.regions = "all arms"` was used, p-value plots can be produced by inserting the `input.regions = "all arms"`, but also for instance `"1p"` or `"20q"`. However, to produce a plot of the whole chromosome, e.g. chromosome 1, the integrated analysis should be re-run with `input.region=1`.

The user may also specify a subset of samples to be considered in the model via the argument `samples` in the function call, which must consist of the list of either column numbers (e.g. `5:ncol(acgh.data)` for both copy number and expression data) or corresponding column names.

SIM allows the incorporation of confounders, such as patient gender, tumor location, tumor type, etc. into the model by using the option `adjust`. Confounder variables can be either continuous or factors, with as many observations as the number of samples on the datasets, and with sample observations in the same order as the samples in the array datasets. See `?integrated.analysis` for details.

Computation of the z-scores can take a long time depending on the datasets' dimensions, and may not be of interest for the entire genome. The default of the `integrated.analysis` function is not to compute it, unless the argument `zscores` is set to `TRUE`.

Let us apply the `integrated.analysis` function to the Pollack copy number and gene expression datasets for chromosome 8q:

```
> integrated.analysis(samples = samples, input.regions = 8, adjust = FALSE,
      zscores = TRUE, method = "auto", run.name = "chr8q")
```

4.4 Plotting and tabulating the P-values

Once the model has been run several functions can be used to obtain overviews of the p-values across the input regions. These functions can also be run when `zcores=FALSE` was used in the `integrated.analysis` function to speed up the analysis.

The first one is `sim.plot.pvals.on.genome`. It will display all multiple-testing corrected p-values according to chromosome location, colored depending on the model outcome (significant or not). Regions rich in statistically significant associations will be mostly blue, while regions with few or no significant associations will be mostly grey. Regions for which the model was not run will be empty. Orange triangles indicate the start and end of the analyzed regions. The purple dot indicates the centromere. The plot will be automatically saved to the run.name directory,unless pdf=FALSE.

```
> sim.plot.pvals.on.genome(input.regions = 8, adjust.method = "BY",
      run.name = "chr8q", pdf = TRUE)

************************************************
The results are stored in the following file:
chr8q/whole_genome_plot.pdf
************************************************
```

Another summary is produced by the function `tabulate.pvals`, which produces tabulations of multiple-testing corrected p-values using cut-offs typically of interest in hypothesis testing. The output is printed on screen. Each row represents an input region with tabulated p-values. The percentage column indicates how many probes in the input region have a p-value below, in this case, the 8th bin (0.2).

```
> tabulate.pvals(input.regions = 8, adjust.method = "BY", bins = c(0.001,
      0.005, 0.01, 0.025, 0.05, 0.075, 0.1, 0.2, 1), significance.idx = 8,
      order.by = "%", decreasing = TRUE, run.name = "chr8q")
```

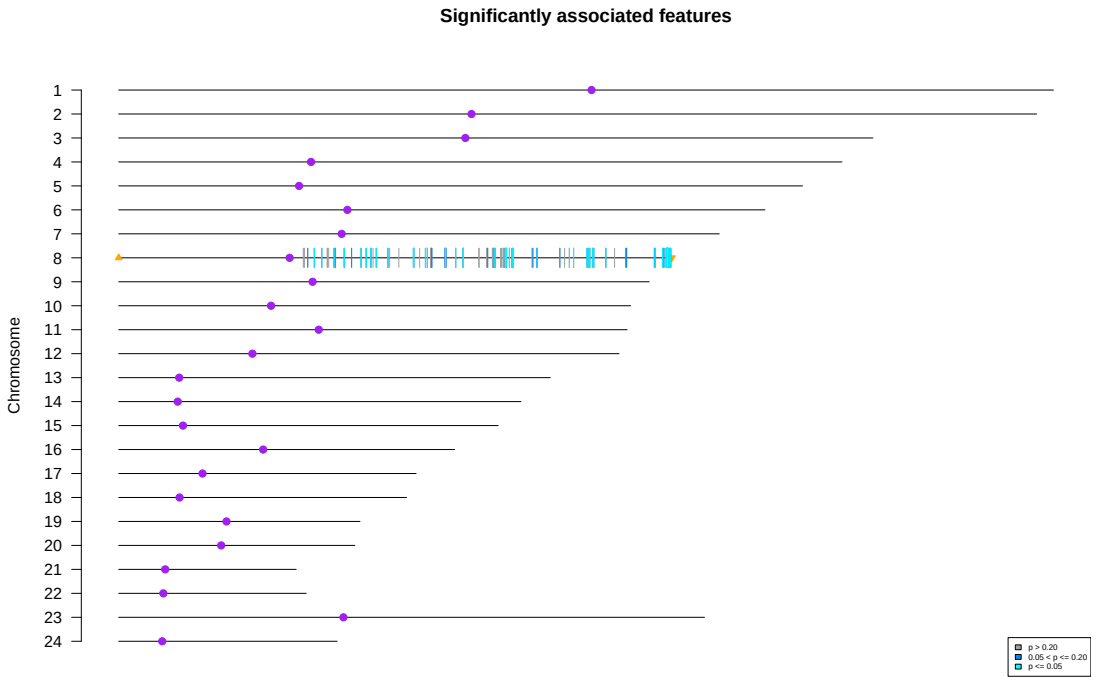

Figure 1: Adjusted p-values on whole genome plot.

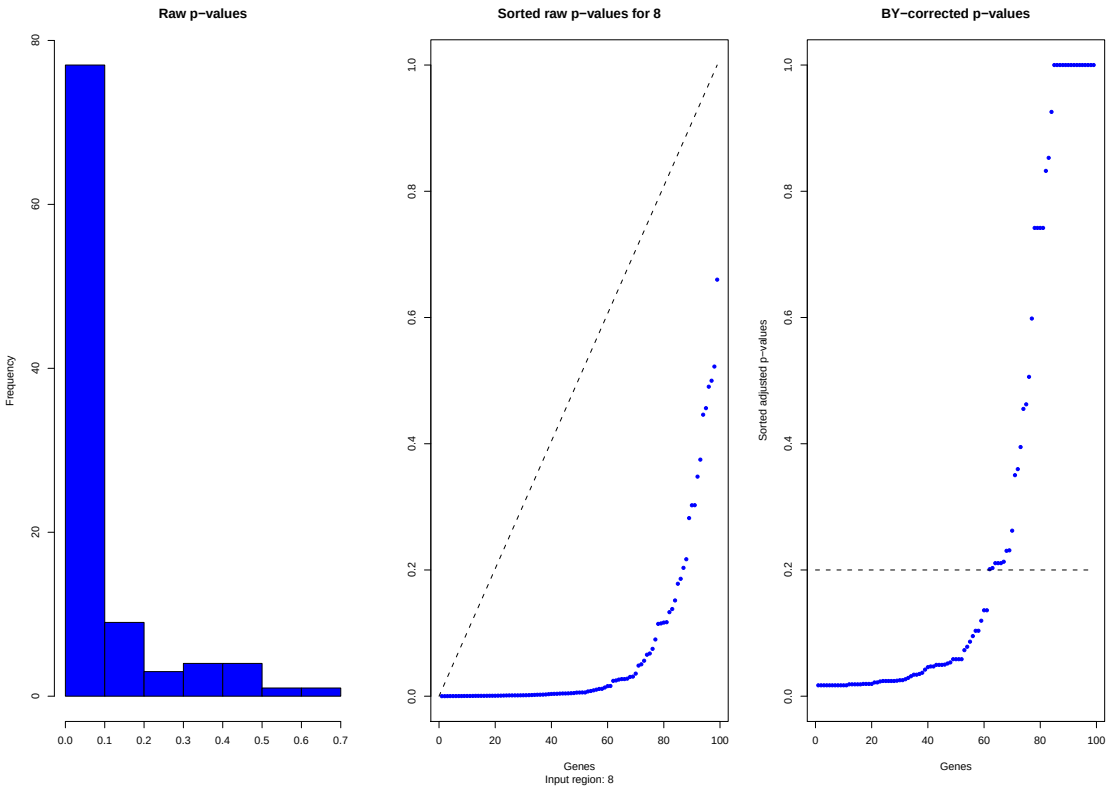

Figure 2: Empirical distribution of the computed p-values.

```
  input.region 0.001 0.005 0.01 0.025 0.05 0.075 0.1 0.2  1        %
1            8     0     0    0    29   45    53  56  61 99 61.61616
```

The function `sim.plot.pvals.on.region` generates a multipage pdf with two pages per analyzed region. The first page contains three plots of the empirical distribution of the computed p-values: a histogram and an empirical cumulative distribution function (c.d.f.) of the uncorrected p-values, as well as the empirical c.d.f. of the multiple-testing corrected p-values. The histogram expected if there is no association between copy number and expression is flat, whilst in the presence of association it will display a higher proportion of small p-values than larger ones. The empirical c.d.f., produced by plotting the sorted p-values, conveys the same information as the histogram: if there is no association, it will produce an approximate diagonal straight line (also plotted as a reference), whilst association will be seen as a convex curve, staying below the expected curve. The empirical c.d.f. of the multiple-testing adjusted p-values is mainly used to visualize how many features would be selected for various thresholds.

The second page contains a plot of the multiple-testing corrected p-values positioned along the genomic region. It is then easy to see if there are sub-regions with mostly low p-values.

```
> sim.plot.pvals.on.region(input.regions = 8, adjust.method = "BY",
      run.name = "chr8q")

************************************************
The results are stored in the following file:
chr8q/pvalue_plots/pvals_on_region.pdf
************************************************
```

By default, the multiple-testing correction used is the false discovery rate (FDR)-controlling method suggested by Benjamini & Yekutiely [2]. It can be used in case the p-values may have been produced by correlated tests, which we believe to be the case while testing association between features located in the same genomic regions. However, the user may also choose to use the less conservative FDR-controlling method suggested by Benjamini & Hochberg [1].

The default Benjamini & Yekutieli multiple testing correction is rather conservative but seems to be the most appropriate when copy number is used as dependent variable. If effects are mild, we suggest increasing the FDR threshold to 20%, and looking for regions rich in corrected p-values below this threshold. Effects are expected to be mild in data sets with fewer than 50 samples, low amplitude changes, and/or copy number changes in a low percentage of samples.

4.5 Visualizing association patterns

The association heatmap can only be drawn when z-scores have been calculated in the `integrated.analysis` (`zscores=TRUE`). The function `sim.plot.zscore.heatmap` produces an association heatmap that shows the association (standardized influence) of each independent feature (expression measurement) with each dependent feature (copy number measurement). The copy number measurements are represented by the rows, whilst the expression measurements are represented by the columns. One heatmap representing each region is produced and stored in the folder `heatmap_zscores`. The copy number probes are on the rows, from start of of region (bottom) to end of the region (top), while the expression probes are on the columns, from start of the region (left) to end of the region (right). Every cell in the heatmap represents association between an individual copy number and expression probe (z-score).

A vertical color bar is added to the left of the association heatmap displaying the discretized adjusted p-values, to help visualize regions where significant association is found. On top, a horizontal color bar is added which indicates genes with mean z-scores above a set threshold across the significant copy number probes. An optional panel gives an overview of the copy number data per sample, representing it either as a heatmap (green for gain/amplification, red for

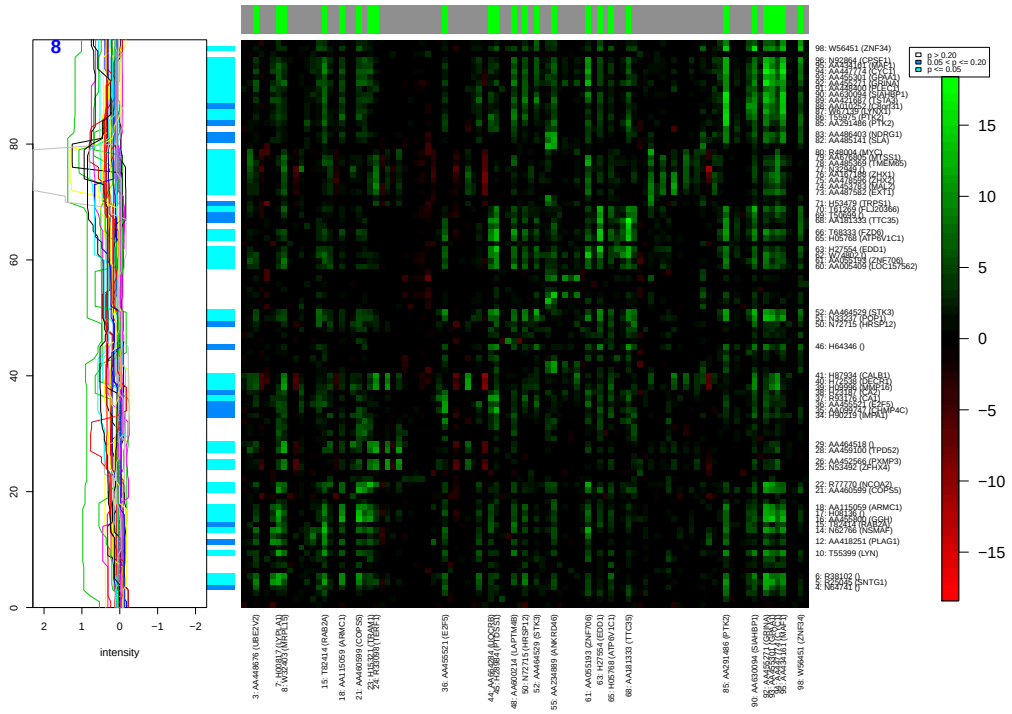

Figure 3: Association heatmap for chromosome 8q with copy number data per sample represents as smoothed medians in the left panel.

loss of genomic DNA), as smoothed medians [3], or as *clac* summary plot [7]. Positive association (green) can mean either copy number gain and increased expression, or deletion and decreased expression.

The heatmaps can also be used in an exploratory analysis, looking for very local effects of copy number changes (usually narrow amplifications) on gene expression, that do not always lead to a significant test result.

Since heatmaps for high-density array platforms can be rather large, it takes some time to produce them. When running multiple chromosomal regions, the default option pdf=TRUE directly saves the graphs into a subdirectory of the run.name folder.

```
> sim.plot.zscore.heatmap(input.regions = 8, significance = 0.2,
      z.threshold = 3, show.names.dep = TRUE, show.names.indep = TRUE,
      adjust.method = c("BY"), scale = "auto", plot.method = "smooth",
      run.name = "chr8q", pdf = TRUE)

Running for input.region: 8

************************************************
The results are stored in the following file:
chr8q/heatmap_zscores/globaltest_heatmap_8_BY_pvals_smooth.pdf
************************************************
```

Note that individual z-scores only depend on the pair of probes involved and, thus, values are exactly the same regardless of the region the model was applied to: the entire genome, a single

chromosome or a sub-chromosomal region. However, in contrast to the z-scores, p-values corresponding to each test will change depending on the region considered.

4.6 Prioritizing candidate regions and genes

The p-values for the copy number probes, representing significance of association with gene expression in the region, are available in a tab-delimited text file for each analyzed region, sorted on significance:

```
> tabulate.top.dep.features(input.regions = 8, adjust.method = "BY",
      run.name = "chr8q")

Creating table for 8

************************************************
The results are stored in the following file:
chr8q/top.dep.features/top.dep.features.8.txt
************************************************

Creating table for 8

************************************************
The results are stored in the following file:
chr8q/top.indep.features/top.indep.features.8.txt
************************************************
```

The genes with the highest mean z-scores across the signficant copy number probes (user defined threshold, here 0.2), can be found in a tab-delimited text file for each analyzed region, sorted on mean z-score.

5 Extensions of the model

5.1 Categorization of copy number data

Both copy number and expression values are taken as continuous variables, so no categorization is done. We believe part of the strength of this approach is to be able to detect associations between subtle changes in copy number and expression, so that categorization is undesirable. Also, without categorization, the actual levels of copy number aberration are taken into account. However, it is possible to use the same model if it is preferable to categorize the data. If for example the copy number is categorized as having either ”change” or ”no change”, the same model with a logistic link could be used.

5.2 Other applications of the model

Our example in section 4 uses the proposed approach to answer the question: which genes have copy number associated with the expression levels of genes on the same chromosomal region? This suggests using copy number as dependent and expression as independent variables.

In other cases, there might be interest in finding genes whose expression levels are associated with copy number changes in and around a fixed region. Expression-regulating mechanisms may involve not only copy number, but also epigenetic and sequence changes, and it may thus be of interest to identify genes whose regulation is closely associated with one of those mechanisms and apply the model to SNP and methylation microarray data.

sessionInfo

The *SIM* package 1.9.0 can be run in R version 2.5 and higher R versions. For this example the following package versions were used:

- R version 2.9.0 (2009-04-17), `x86_64-unknown-linux-gnu`
- Locale: `LC_CTYPE=en_US;LC_NUMERIC=C;LC_TIME=en_US;LC_COLLATE=en_US;LC_MONETARY=C;LC_MESSAGES=e`
- Base packages: base, datasets, graphics, grDevices, grid, methods, stats, tools, utils
- Other packages: Biobase 2.4.0, clac 0.1-2, fields 5.02, globaltest 4.14.0, limma 2.18.0, lodplot 1.1, marray 1.22.0, multtest 2.0.0, quantreg 4.27, quantsmooth 1.10.0, SIM 1.12.0, spam 0.15-4, SparseM 0.79
- Loaded via a namespace (and not attached): annotate 1.22.0, AnnotationDbi 1.6.0, DBI 0.2-4, MASS 7.2-46, RSQLite 0.7-1, splines 2.9.0, survival 2.35-4, xtable 1.5-5

Acknowledgements

We are very grateful to Jelle Goeman for many helpful discussions, and to the contributions of Olga Tsoi and Nina Tolmacheva to the first version of the package. This work was conducted within the Centre for Medical Systems Biology (CMSB), established by the Netherlands Genomics Initiative/Netherlands Organisation for Scientific Research (NGI/NWO). This work has been partially supported by the project BioRange of The Netherlands Bioinformatics Centre (NBIC).

References

[1] Y Benjamini and Y Hochberg. Controlling the false discovery rate – a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society Series B-Methodological*, 57:289–300, 1995.

[2] Y Benjamini and D Yekutieli. The control of the false discovery rate in multiple testing under dependency. *Annals of Statistics*, 29:1165–1188, 2001.

[3] PHC Eilers and RX de Menezes. Quantile smoothing of array cgh data. *Bioinformatics*, 21:1146–1153, 2005.

[4] JJ Goeman, SA van de Geer, F de Kort, and HC van Houwelingen. A global test for groups of genes: testing association with a clinical outcome. *Bioinformatics*, 20:93–09, 2004.

[5] RX Menezes, M Boetzer, M Sieswerda, GJ van Ommen, and JM Boer. Integrated statistical analysis to identify associations between dna copy number and gene expression in microarray data. *Submitted.*, 2008.

[6] JR Pollack, T Sorlie, CM Perou, CA Rees, SS Jeffrey, PE Lonning, R Tibshirani, D Botstein, AL Borresen-Dale, and PO Brown. Microarray analysis reveals a major direct role of dna copy number alteration in the transcriptional program of human breast tumors. *Proceedings of the National Academy of Sciences of the United States of America*, 99:12963–12968, 2002.

[7] P Wang, Y Kim, J Pollack, B Narasimhan, and R Tibshirani. A method for calling gains and losses in array cgh data. *Biostatistics*, 6:45–58, 2005.