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 1.62.0
R.Menezes@VUmc.NL

May 19, 2021

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* and *quantsmooth* Bioconductor packages.

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. This is more efficient than keeping large objects on the workspace, 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 and 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 (X and Y for the sex-chromosomes), 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 of type `data.frame`, and they include the probe annotations. To find out which columns they contain, use

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

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

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

```
 [1] "ID"          "Symbol"
 [3] "CHROMOSOME"  "STARTPOS"
 [5] "Abs.start"   "BT474"
 [7] "MCF7"        "NORWAY.10"
 [9] "NORWAY.100"  "NORWAY.101"
[11] "NORWAY.102"  "NORWAY.104"
[13] "NORWAY.109"  "NORWAY.11"
[15] "NORWAY.111"  "NORWAY.112"
[17] "NORWAY.12"   "NORWAY.14"
[19] "NORWAY.15"   "NORWAY.16"
[21] "NORWAY.17"   "NORWAY.18"
[23] "NORWAY.19"   "NORWAY.26"
[25] "NORWAY.27"   "NORWAY.39"
[27] "NORWAY.41"   "NORWAY.47"
[29] "NORWAY.48"   "NORWAY.53"
[31] "NORWAY.56"   "NORWAY.57"
[33] "NORWAY.61"   "NORWAY.65"
[35] "NORWAY.7"    "SKBR3"
[37] "STANFORD.14" "STANFORD.16"
[39] "STANFORD.17" "STANFORD.2"
[41] "STANFORD.23" "STANFORD.24"
[43] "STANFORD.35" "STANFORD.38"
[45] "STANFORD.A"  "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 + base pair position.

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,
                dep.ann = colnames(acgh.data)[1:4],
                indep.ann = 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")
```

```
Assembling data ...
... assembled dependent data: dim(99, 46).
... assembled independent data: dim(99, 46).
```

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 `globaltest` can calculate the p-values using two different models, using a asymptotic distribution or based on permutations. The asymptotic model is recommended for large sample sizes but can be conservative for small sample sizes and permutations are recommended for smaller sample sizes or when the asymptotic distribution cannot be assumed. When confounders are included in the model, it is not possible to use the permutation method, so the asymptotic method is suggested. By default the asymptotic is used, if `permutation = nperm`, where $nperm > 0$ is applied as arguments thena the permutation model is used.

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 `"1 1-1000000"` or `"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` e.g. `adjust= gender`. 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=TRUE`.

The integrated analysis can be run using of of the following methods: a) "full"; the full indepedent data will be used as a gene set for the `globaltest`, b) "overlap"; only the gene/probes of the independent datat that overlap with the dependent data will be used, c) "smooth"; the dependent data will be smoothed using `quantsmooth`, d) "window"; only the independent genes/probes that fall in a window around the dependent genes/probes are used as a gene set for the `globaltest`. Additional the window-size and end-position of the dependent genes/probes can be given.

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 = "8q",
                      zscores = TRUE,
                      method = "full",
                      run.name = "chr8q")
```

```
Performing integrated analysis on input region(s): 8q.
... input region: 8q
    ... features 4 of 99 (19s)
    ... features 8 of 99 (10s)
    ... features 12 of 99 (10s)
    ... features 16 of 99 (8s)
    ... features 20 of 99 (8s)
    ... features 24 of 99 (9s)
    ... features 28 of 99 (8s)
    ... features 32 of 99 (7s)
    ... features 36 of 99 (6s)
    ... features 40 of 99 (7s)
    ... features 44 of 99 (6s)
    ... features 48 of 99 (5s)
    ... features 52 of 99 (5s)
    ... features 56 of 99 (5s)
    ... features 60 of 99 (4s)
    ... features 64 of 99 (4s)
    ... features 68 of 99 (3s)
    ... features 72 of 99 (3s)
    ... features 76 of 99 (2s)
    ... features 80 of 99 (2s)
    ... features 84 of 99 (2s)
    ... features 88 of 99 (1s)
    ... features 92 of 99 (1s)
    ... features 96 of 99 (0s)
    ... summary results of 'integrated.analysis()' on last feature:
     ... "gt.object" object from package globaltest
     ... Call:
     ...  gt(response = y, alternative = x, null = adjust)
     ... Model: linear regression.
     ... Degrees of freedom: 41 total; 1 null; 1 + 99 alternative.
     ... Null distibution:  asymptotic.
```

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. The organe line indicates the analyzed regions. The purple dots indicates the centromere. The plot will be automatically saved to the "run.name" directory, unless `pdf=FALSE`.

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

Producing genome plot ...
... input region: 8q
... overlapping plot stored with file name:
chr8q/pvalue.plots/WholeGenomePlotfullBY.pdf
```

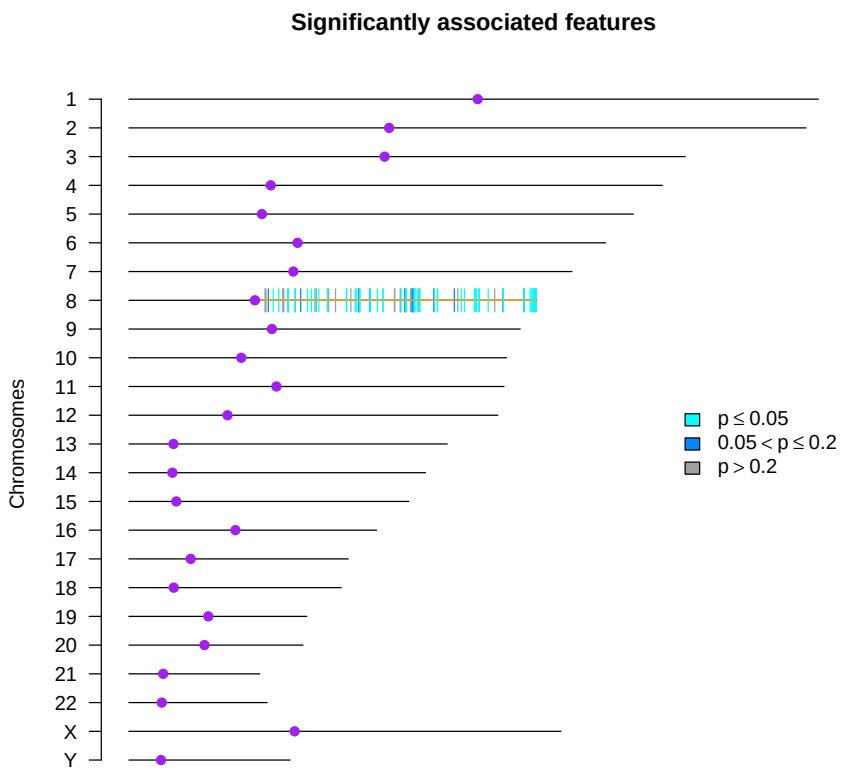

Figure 1: Adjusted p-values on whole genome plot.

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 = "8q",
                 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,
                 run.name = "chr8q")
```

```
Tabulate P-values ...
... input region: 8q
  input.region 0.001 0.005 0.01 0.025
1           8q     2    23   35    52
  0.05 0.075 0.1 0.2  1 8th (%)
1   56    61  61  70 99   70.71
```

The function `sim.plot.pvals.on.region` generates a multipage pdf one page per analyzed region. the first three panels describe 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

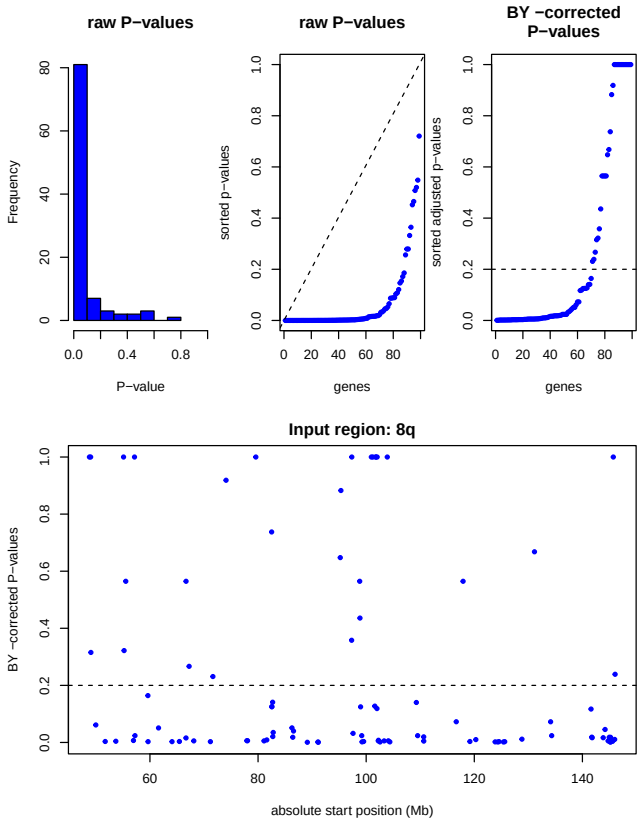

Figure 2: Empirical distribution of the computed p-values.

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 lower panel 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 = "8q", adjust.method = "BY", run.name = "chr8q")

Producing region plot ...
... input region: 8q
... region plot stored with file name:
chr8q/pvalue.plots/PvalueOnRegionfullBY2e-01.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 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).

To indicate the significant copy number measurements (p-value ≤ 0.2) a yellow box is draw in place of the significant row left and right to the association heatmap. The significant z-scores are indicated by the colours given by the colour key below the associated heatmap, in this case dark blue indicates positive z-scores. An optional panel gives an overview of the copy number data per sample, representing it either as a heatmap or as smoothed medians [3].

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.

In this example we use the *RColorBrewer* package for a nice diverging colour palette. We also adapted the scale of the added smoothed medians plot.

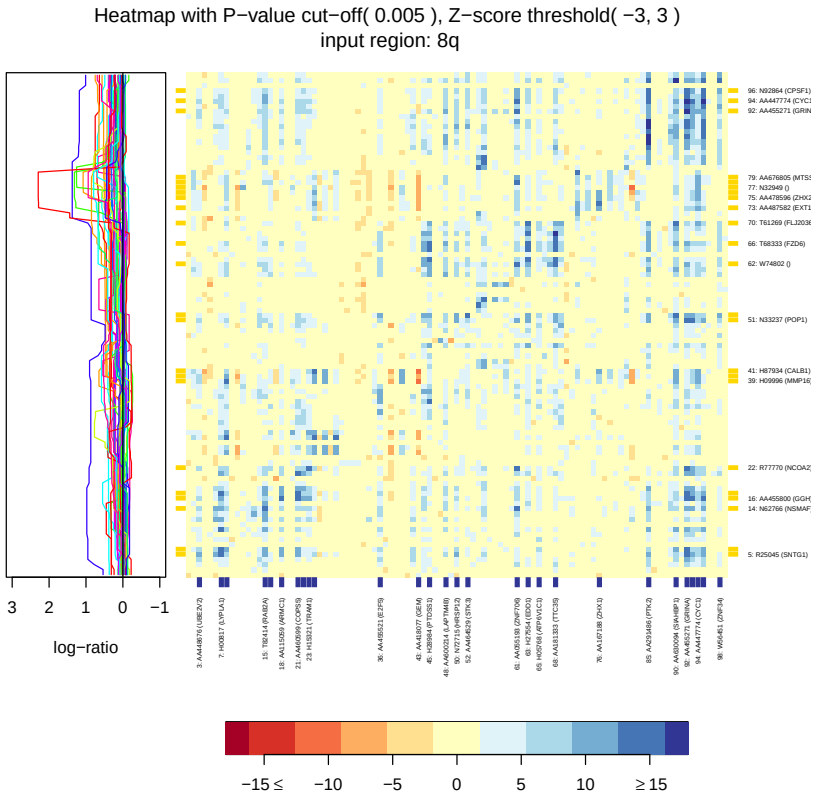

Figure 3: Association heatmap for chromosome 8q with copy number data per sample represents as smoothed medians in the left panel.

```
> library(RColorBrewer)
> sim.plot.zscore.heatmap(input.regions="8q",
                          method="full",
                          significance=0.005,
                          z.threshold=3,
                          colRamp=brewer.pal(11, "RdYlBu"),
                          add.colRamp=rainbow(10),
                          show.names.indep=TRUE,
                          show.names.dep=TRUE,
                          adjust.method="BY",
                          add.plot="smooth",
                          add.scale=c(-1, 3),
                          pdf=TRUE,
                          run.name="chr8q")

Producing heatmap ...
... input region: 8q
... heatmap plot stored with file name:
chr8q/heatmap.zscores/Heatmap8qfullsmoothBY5e-03.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 or as a `list` of `data.frame`'s for each input region:

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

Tabulate results on dependent data ...
... input region: 8q
... tabulated P-values stored with file name:
chr8q/top.dep.features/TopDepFeatures8qfullBY1e+00.txt

> head(table.dep[["8q"]])

   P.values extreme.influences
39  0.00063              11.50
41  0.00063              11.48
40  0.00160              10.95
92  0.00160              15.92
94  0.00160              16.58
62  0.00170              13.91
   absolute.start.position       ID
39              8089118578   H09996
41              8091140007   H87934
40              8091082756   H72538
92              8145136277 AA455271
94              8145221980 AA447774
62              8102572572   W74802
   Symbol CHROMOSOME  STARTPOS     ID.1
39  MMP16          8  89118578   W32403
41  CALB1          8  91140007   R33098
40  DECR1          8  91082756   H72538
92  GRINA          8 145136277 AA455271
94   CYC1          8 145221980 AA455271
62                 8 102572572 AA055193
   Symbol.1 CHROMOSOME.1 STARTPOS.1
39   MRPL15            8   55222599
41    TERF1            8   74083648
40    DECR1            8   91082756
92    GRINA            8  145136277
94    GRINA            8  145136277
62   ZNF706            8  102278381

Tabulate results on independent data ...
... input region: 8q
... tabulated results stored with file name:
chr8q/top.indep.features/TopIndepFeatures8qfullBY1e+00.txt
```

```
   P.values extreme.influences
39  0.00063              11.50
41  0.00063              11.48
40  0.00160              10.95
92  0.00160              15.92
94  0.00160              16.58
62  0.00170              13.91
   absolute.start.position       ID
39              8089118578   H09996
41              8091140007   H87934
40              8091082756   H72538
92              8145136277 AA455271
94              8145221980 AA447774
62              8102572572   W74802
   Symbol CHROMOSOME  STARTPOS     ID.1
39  MMP16          8  89118578   W32403
41  CALB1          8  91140007   R33098
40  DECR1          8  91082756   H72538
92  GRINA          8 145136277 AA455271
94   CYC1          8 145221980 AA455271
62                 8 102572572 AA055193
   Symbol.1 CHROMOSOME.1 STARTPOS.1
39   MRPL15            8   55222599
41    TERF1            8   74083648
40    DECR1            8   91082756
92    GRINA            8  145136277
94    GRINA            8  145136277
62   ZNF706            8  102278381
```

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.

```
> sim.plot.overlapping.indep.dep.features(input.regions="8q",
                                          adjust.method="BY",
                                          significance=0.1,
                                          z.threshold= c(-1,1),
                                          log=TRUE,
                                          summarize="consecutive",
                                          pdf=TRUE,
                                          method="full",
                                          run.name="chr8q")
```

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

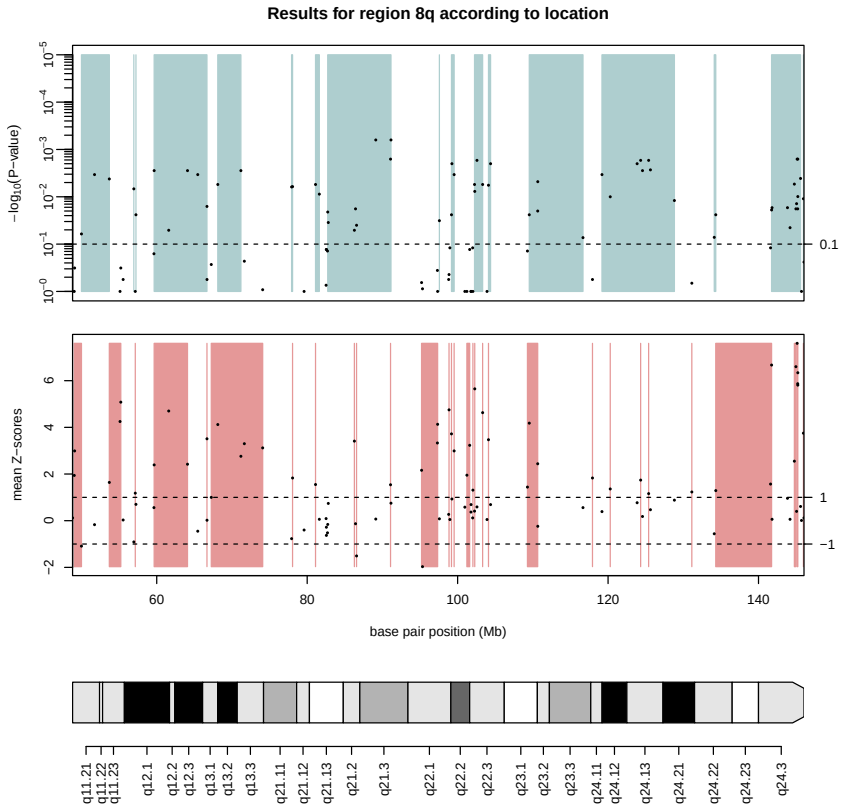

Figure 4: Showing consecutive regions in the dependent and independent data.

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 1.62.0 can be run in R version 2.9 and higher R versions. For this example the following package versions were used:

- R version 4.1.0 RC (2021-05-10 r80283), `x86_64-apple-darwin17.0`
- Locale: `C/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8`
- Running under: `macOS Mojave 10.14.6`
- Matrix products: default
- BLAS: `/Library/Frameworks/R.framework/Versions/4.1/Resources/lib/libRblas.dylib`
- LAPACK: `/Library/Frameworks/R.framework/Versions/4.1/Resources/lib/libRlapack.dylib`
- Base packages: base, datasets, grDevices, graphics, methods, stats, utils
- Other packages: RColorBrewer 1.1-2, SIM 1.62.0, SparseM 1.81, quantreg 5.85
- Loaded via a namespace (and not attached): AnnotationDbi 1.54.0, Biobase 2.52.0, BiocGenerics 0.38.0, Biostrings 2.60.0, DBI 1.1.1, GenomeInfoDb 1.28.0, GenomeInfoDbData 1.2.6, IRanges 2.26.0, KEGGREST 1.32.0, Matrix 1.3-3, MatrixModels 0.5-0, R6 2.5.0, RCurl 1.98-1.3, RSQLite 2.2.7, Rcpp 1.0.6, S4Vectors 0.30.0, XML 3.99-0.6, XVector 0.32.0, annotate 1.70.0, bit 4.0.4, bit64 4.0.5, bitops 1.0-7, blob 1.2.1, cachem 1.0.5, compiler 4.1.0, conquer 1.0.2, crayon 1.4.1, fastmap 1.1.0, globaltest 5.46.0, grid 4.1.0, httr 1.4.2, lattice 0.20-44, matrixStats 0.58.0, memoise 2.0.0, parallel 4.1.0, png 0.1-7, quantsmooth 1.58.0, rlang 0.4.11, rstudioapi 0.13, splines 4.1.0, stats4 4.1.0, survival 3.2-11, tools 4.1.0, vctrs 0.3.8, xtable 1.8-4, zlibbioc 1.38.0

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. *BMC Bioinformatics.*, 10:203, 2009.

[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.