Testing association of a pathway with a clinical variable

Jelle Goeman Jan Oosting

December 14, 2005

Contents

1 Introduction

This document shows the functionality of the R-package *globaltest*. The main function tests whether a given group of genes is significantly associated with a clinical outcome. The explanation here focuses on practical use of the test. To understand the idea and the mathematics behind the test, and for more details on how to interpret a test result, we refer to the papers (Goeman et al. 2004, 2005).

In the last few years there has been a shift in focus from studying the effects of single genes to studying effects of multiple functionally related genes. Most of the current methods for studying pathways involve looking at increased proportions of differentially expressed genes in pathways of interest. These methods do not identify pathways where many genes have altered their expression in a small way. The globaltest was designed to address this issue.

Essentially the globaltest is based on an empirical Bayesian generalized linear model where the regression coefficients between expression data and clinical outcome are the random variables. The test applies a goodness of fit test on this model. The globaltest is optimal when all coefficients show an effect, even if none of the individual coefficients shows a significant relation between expression and outcome. However the null hypothesis will also be rejected when there are 1 or a few coefficients with a significant relation.

By grouping together genes before testing usually the number of tests will decrease and amount of correction for multiple testing will be decreased. Genes can be grouped together in any meaningful way, for example based in function (KEGG, GO) or location (chromosome, cytoband).

In our examples we use data sets that are available from the BioConductor web site. All the necessary packages to repeat the examples below are available from www.bioconductor.org.

First we load some packages from BioConductor and load the data we'll use.

```
> library(globaltest)
> library(golubEsets)
> library(vsn)
> data(golubMerge)
> golubM <- update2MIAME(golubMerge)
> golubX <- vsn(golubM)
```

This gives us a data set `golubX`, which is of the format *exprSet*, the standard format for gene expression data in BioConductor. It has 7,129 genes for 72 samples. We used *vsn* to normalize the data. Any other normalization method may be used instead. Several phenotype variables are available with `golubX`, among them "ALL.AML", the clinical variable that interests us.

In this document we use the globaltest based on BioConductor *exprSet* input. For examples using simple vector or matrix input, see `help(globaltest)`.

2 Global Testing of a Single Pathway

Suppose we are interested in testing whether AML and ALL have different gene expression pattern for certain pathways from the KEGG database.

First we load all KEGG and GO pathways. This is most easily done using the various metadata annotation packages that are available on BioConductor. For the Golub data, which were made using an Affymetrix hu6800 chip, we use the *hu6800* metadata package. If no metadata package is available for the chip or organism used, users can make one with the *AnnBuilder* package.

```
> library(hu6800)
> kegg <- as.list(hu6800PATH2PROBE)
> go <- as.list(hu6800GO2ALLPROBES)
```

This creates a list `kegg` of 153 pathways and a list `go` of 4979 pathways. Each pathway in these lists is a vector of gene names.

In GO, it is sometimes wise to only consider annotations which were curated by experts and to exclude those inferred by electronic annotation. Excluding the electronic and unspecified GO-annotations can be done with

```
> go <- lapply(go, function(x) x[!is.na(names(x)) & (names(x) !=
+     "IEA")])
```

We can now retrieve the Cell Cycle genes according to these two annotations.

```
> cellcycleGO <- go[["GO:0007049"]]
> cellcycle <- kegg[["04110"]]
```

The vectors `cellcycle` (92 probes) and `cellcycleGO` (330 probes) contain the probes on the hu6800 chip that are annotated to the Cell Cycle pathway. Suppose we are predominantly interested in this pathway. We want to know whether this group of genes is associated with the clinical outcome AML versus ALL.

2.1 Testing the pathway of all genes

Always first test all genes to see if the overall gene expression pattern is different for different clinical outcomes. We can do this by saying

```
> gt.all <- globaltest(golubX, "ALL.AML")
```

The first input X should be the *exprSet* object, the second input Y the name of the clinical variable in `pData(X)`. Alternatively we can give a matrix of expressions as X and a vector as Y.

The test result is stored in a *gt.result* object, which also contains all the information needed to draw the plots.

```
> gt.all
```

```
Global Test result:
Data: 72 samples with 7129 genes; 1 pathway tested
Model: logistic

    genes tested Statistic Q Expected Q sd of Q    P-value
all  7129   7129      53.992         10  1.9035 5.1617e-35
```

We conclude that there is ample evidence that the overall gene expression profile for all 7,129 genes is associated with the clinical outcome: samples with similar AML/ALL status tend to have similar expression profiles. In cases such as this one, in which the overall expression pattern is associated with the clinical variable, we can expect most pathways (especially the larger ones) also to be associated with it.

Because `golubX` is an *exprSet*, we could simply give the name of the phenotype variable "AML.ALL" as our Y input. Alternatively, we can give a vector here.

2.2 Different models

The Global Test allows three different kinds of clinical variables to be tested.

- A clinical variable defining two groups, i.e. having two values (using the logistic model). For a multi-valued clinical variable, the option *levels* can be used to set which groups are to be tested against each other.
- A continuously distributed measurement (using the linear model).
- A survival time (using the Cox model). In that case Y should contain the last observation time of each individual, and an extra argument d should be supplied which contains the event indicator, which has value *event* if an event occurred.

The function `globaltest` will automatically choose an appropriate model based on Y. To override the automatic choice, use the option *model*.

The syntax in case of a survival model is as follows. The `pData` object of the *exprSet* `X` object should contain a variable with the survival times (say "time"), and a variable with the status (say "status"), in which 1 indicates that the subject died and 0 that it was censored. `globaltest` can then be called with either

```
> gt <- globaltest(X, "Surv(time, status)")
> gt <- globaltest(X, "time", d = "status")
```

For more information see the function `Surv` in the *survival* package.

2.3 Testing a single pathway

Now we test the Cell Cycle pathway that interests us:

```
> globaltest(golubX, "ALL.AML", cellcycleGO)

Global Test result:
Data: 72 samples with 7129 genes; 1 pathway tested
Model: logistic

     genes tested Statistic Q Expected Q sd of Q    P-value
[1,]   330    250      79.257     12.389    3.27 1.5999e-24

> gt.cc <- globaltest(golubX, "ALL.AML", cellcycle)
> gt.cc
```

```
Global Test result:
Data: 72 samples with 7129 genes; 1 pathway tested
Model: logistic

     genes tested Statistic Q Expected Q sd of Q    P-value
[1,]    92     92      69.005     10.195  3.2811 1.4669e-18
```

Note from the results of the GO cell cycle pathway that `globaltest` automatically throws away duplicate probe ids. The vector `cellcycleGO` is of length 330, but only contains 250 unique probe ids.

We conclude from the test results that the expression pattern of the cellcycle pathway is notably different between AML and ALL samples. However, as the test on all genes was significant we can generally expect most pathways to be significant as well. To get an impression of how "special" this pathway is compared to other pathways in this data set, one can use the function `sampling`.

```
> gt.cc <- sampling(gt.cc)
> gt.cc

Global Test result:
Data: 72 samples with 7129 genes; 1 pathway tested
Model: logistic

     genes tested Statistic Q Expected Q sd of Q    P-value Comparative p
[1,]    92     92      69.005     10.195  3.2811 1.4669e-18         0.244
```

This gives an extra output column "Comparative p", which is the fraction of random genesets of the same size as the cell cycle pathway (92 genes) which have a lower p-value than cell cycle itself. In this case around 24 % of 1,000 random 'pathways' of size 92 have a lower p-value than the Cell Cycle pathway. By default 1,000 random sets are sampled; this number can be changed with the option *ndraws*.

By default the p-value of globaltest is calculated using approximate formulas which are more accurate for large sample size, but may be inaccurate for very small sample size. For 72 arrays they should be accurate enough. For very small sample sizes an alternative is to use the permutation version of globaltest. This recalculates the p-value on the basis of 10,000 permutations of the clinical variable.

```
> permutations(gt.cc)

Global Test result:
Data: 72 samples with 7129 genes; 1 pathway tested
Model: logistic
Using 10000 permutations of Y

     genes tested Statistic Q Expected Q sd of Q P-value Comparative p
[1,]    92     92      69.005     10.351  3.3452       0         0.244
```

The permutation p-value is not so accurate in the lower range as it is always a multiple of one over the number of permutations and also has some sampling variance. If desired, the number of permutations can be changed with the option *nperm* to get more accurate p-values.

2.4 Adjusting for the presence of covariates

It is also possible to adjust the globaltest for confounders or for known risk factors. For example in the Golub Data set we may be afraid for a disturbance due to that the fact that some samples were taken from peripheral blood while others were taken from bone marrow. We can correct for this using the option *adjust*. The option *adjust* can also be used when the study design is different from the simple 'two independent samples' design of the standard global test. In a paired design, for example, put the pair-id (as a factor) in *adjust*.

The user may supply one or more names of covariates in the option *adjust* or supply *adjust* as a *data.frame*. The easiest way of adjustment, however, is by using a *formula* object as input for Y, as follows:

```
> globaltest(golubX, ALL.AML ~ BM.PB, cellcycle)

Global Test result:
Data: 72 samples with 7129 genes; 1 pathway tested
Model: logistic, ALL.AML ~ BM.PB
Adjusted: 99.8 % of variance of Y remains after adjustment

     genes tested Statistic Q Expected Q sd of Q    P-value
[1,]    92     92      69.358     10.125   3.302 2.0127e-18
```

The test result now also gives the percentage of the variance in Y that was left after the adjustment. It is a crude measure like $1 - R^2$. If the percentage is low, the adjustment already explained most of the variance of the outcome Y and there was not much residual variance left to test the influence of the genes. To see an example, adjust for "Source" instead of "BM.PB".

The option *adjust* may again be combined with the function *sampling*, but not with *permutation*.

3 Multiple Global Testing

It is also possible to test many pathways at once. To test all KEGG pathways we proceed as follows:

```
> gt.kegg <- globaltest(golubX, "ALL.AML", kegg)
```

The result `gt.kegg` can be displayed and prints a matrix whose rows correspond to the KEGG pathways. It gives the test results for each pathway. We can also display only some of them:

```
> gt.kegg[1:10]

Global Test result:
Data: 72 samples with 7129 genes; 10 pathways tested
Model: logistic

      genes tested Statistic Q Expected Q sd of Q    P-value
00970    23     23     40.5300     9.3642  5.9616 6.5427e-04
00100    13     13     76.0750    10.3220  6.2824 2.5998e-07
```

```
05020    12     12      9.3932     7.5489  3.5372 2.6169e-01
01510    50     50     21.4750     8.8044  2.6204 1.5510e-04
00430     8      8     42.4510     7.0544  5.6346 3.4005e-04
00190    65     65     45.2870    10.4110  4.8683 9.3723e-06
04510   234    234     48.8730     8.9377  1.9326 3.6071e-28
00650    41     41     32.3360     9.5021  4.6725 5.4420e-04
04512    99     99     30.9100     9.0156  2.1169 1.4009e-11
04620    96     96     90.0520    11.8950  4.4522 1.7682e-16
```

```
> gt.kegg["04110"]

Global Test result:
Data: 72 samples with 7129 genes; 1 pathway tested
Model: logistic

      genes tested Statistic Q Expected Q sd of Q    P-value
04110    92     92      69.005     10.195  3.2811 1.4669e-18
```

The same options described above for the single pathway `globaltest` can be applied to the multiple pathway version of `globaltest` as well.

Two functions allow further processing to be done on the test results. The function `result` extracts the whole matrix of test results, while the function `p.value` only extracts the vector of p-values. The latter function can be used for example when a correction for multiple testing is to be done. Note however that due to the extremely high correlations between the tests for different pathways, many multiple testing procedures are inappropriate for the Global Test. See the *multtest* package for details.

We might want to sort the pathways by their p-value, and show the top five. This can be done as follows

```
> sort.gt.kegg <- sort(gt.kegg)
> sort.gt.kegg[1:5]

Global Test result:
Data: 72 samples with 7129 genes; 5 pathways tested
Model: logistic

      genes tested Statistic Q Expected Q sd of Q    P-value
04060   243    243      77.215     9.9820  2.7175 5.2414e-30
04610    81     81     113.350     8.8611  3.4384 2.8036e-29
04510   234    234      48.873     8.9377  1.9326 3.6071e-28
04640   119    119     159.390    17.0210  5.9894 5.2684e-24
00590    30     30     215.300    13.7490  6.8877 3.2333e-23
```

To make this list more readable, the pathway numbers can be replaced by the pathway names:

```
> library(KEGG)
> names(sort.gt.kegg) <- as.list(KEGGPATHID2NAME)[names(sort.gt.kegg)]
> sort.gt.kegg[1:5]
```

```
Global Test result:
Data: 72 samples with 7129 genes; 5 pathways tested
Model: logistic
```

	genes	tested	Statistic Q	Expected Q
Cytokine-cytokine receptor interaction	243	243	77.215	9.9820
Complement and coagulation cascades	81	81	113.350	8.8611
Focal adhesion	234	234	48.873	8.9377
Hematopoietic cell lineage	119	119	159.390	17.0210
Prostaglandin and leukotriene metabolism	30	30	215.300	13.7490

	sd of Q	P-value
Cytokine-cytokine receptor interaction	2.7175	5.2414e-30
Complement and coagulation cascades	3.4384	2.8036e-29
Focal adhesion	1.9326	3.6071e-28
Hematopoietic cell lineage	5.9894	5.2684e-24
Prostaglandin and leukotriene metabolism	6.8877	3.2333e-23

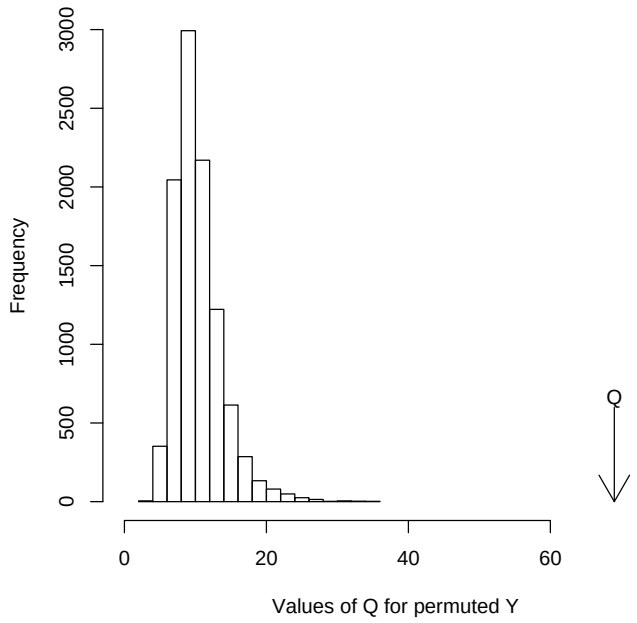
4 Diagnostic plots

There are various types of diagnostic plots available to help the user interpret the `globaltest` result. The plot `permutations` can serve as a check whether the sample size was large enough not to use the permutation version of `globaltest`. The `geneplot` visualizes the influence of individual genes on the test result. The three plots `sampleplot`, `checkerboard` and `regressionplot` all visualize the influence of individual samples. Of these three, `sampleplot` is probably the most useful.

4.1 Permutations histogram

The permutations histogram plots the values of the test statistic Q calculated for permutations of the clinical outcome in a histogram. The observed value of Q for the true values of the clinical outcome is marked with an arrow.

```
> hist(permutations(gt.cc))
```



The output can be interpreted as a plot of the distribution of the test statistic under the null hypothesis that the pathway is not associated with the clinical variable. Strictly speaking, however, the permutation version of the Global Test is a different test with different properties (especially for survival data). It may give different p-values for small samples.

The function `permutations` may not be used when the adjusted version of `globaltest` was used.

4.2 Gene Plot

The second diagnostic plot is the Gene Plot, which can be used to assess the influence of each gene on the outcome of the test. The Gene Plot gives a bar and a reference line for each gene tested. The bar indicates the influence of each gene on the test statistic.

A reference line for each bar gives the expected height of the bar under the null hypothesis that the gene is not associated with the clinical outcome (except in a survival model, where the expected height is zero). Marks indicate with how many standard deviations (under the null) the bar exceeds the reference line. Finally the bars are coloured to indicate a positive or a negative association of the gene with the clinical outcome.

The geneplot bars have two interpretations. In the first place, the bars are the Global Test statistic for the single gene pathway containing only that gene. A positive bar that is many standard deviations above the reference line therefore indicates a gene that is significantly associated with the clinical variable in Y. Secondly, the bars indicate the influence of the gene on the test result of the whole pathway (the test statistic for the group is the average of the bars for

the genes). Removing a gene with a low bar (relative to the reference line) or a negative bar from the pathway will result in a lower p-value for the pathway, removing a gene with a tall positive bar will have the opposite effect.

To plot the geneplot, use any of the commands below:

```
> geneplot(gt.cc)
> geneplot(gt.kegg, "04110")
> geneplot(gt.kegg["04110"])
```

For a large number of genes the plot might become overcrowded. Use the option *genesubset* to plot only a subset of the genes, *labelsize* to resize the gene labels or *drawlabels = FALSE* to remove them. Alternatively, we can plot part of the geneplot later, as follows

```
> gp.cc <- geneplot(gt.cc)
> plot(gp.cc[1:40])
```

This allows one to look at subsets of a large pathway more closely. The return of the `geneplot` is an object of type *gt.barplot* containing the numbers and names appearing in the plot:

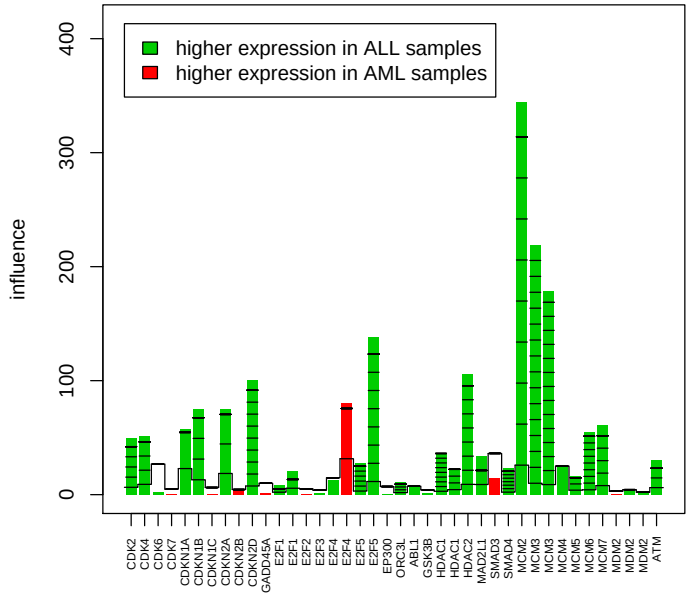
```
> gp.cc[1:10]
```

	Influence	Expected	SD	z.score	colouring
M68520_at	49.7145218	6.441802	8.919950	4.851228948	high in ALL
U37022_rna1_at	51.3140337	9.032787	12.431882	3.401033367	high in ALL
X66365_at	2.3985950	26.901314	36.936158	-0.663380274	high in ALL
L20320_at	0.4454402	4.832082	6.743828	-0.650467543	high in AML
U09579_at	57.1091980	22.869922	31.725262	1.079243284	high in ALL
HG4258-HT4528_at	74.8159314	13.040274	18.168260	3.400196779	high in ALL
U22398_at	0.1904305	6.247576	8.688779	-0.697122799	high in AML
U26727_at	75.1724653	18.584538	25.895921	2.185206190	high in ALL
L36844_at	4.5553260	4.493117	6.247460	0.009957535	high in AML
U40343_at	100.1878850	7.591272	10.507570	8.812371533	high in ALL

For interpretability, the probe identifiers appearing in the plot can be replaced by the gene symbols:

```
> names(gp.cc) <- as.list(hu6800SYMBOL)[names(gp.cc)]
> plot(gp.cc[1:40])
```

The option *scale* can be used to rescale the bars to have unit standard deviation. Alternatively, this can be done later with the command `scale`.



4.3 Sample Plot

The Sample Plot looks very similar to the Gene Plot and visualizes the influence of the individual samples on the test result. It has a bar and a reference line for each sample tested. The bar indicates the influence of each sample on the test statistic, similar to the `geneplot`. The direction of the bar (upward or downward) indicates evidence against or in favour of the null hypothesis. If a sample has a positive bar, its expression profile is relatively similar to that of samples which have the same value of the clinical variable and relatively unlike the profile of the samples which have a different value of the clinical variable. If the bar is negative, it is the other way around: the sample is more similar in expression profile to samples with a different clinical variable. A small p-value will therefore generally coincide with many positive bars. If there are still tall negative bars, these indicate deviating samples: removing a sample with a negative bar would result in a lower p-value.

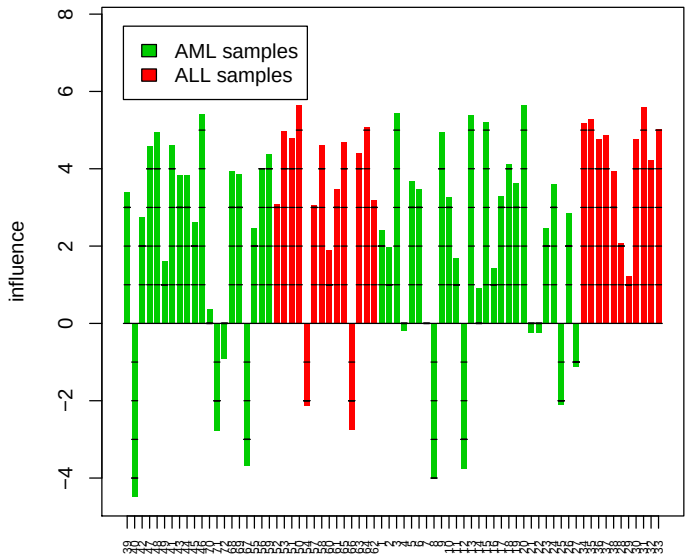
If the null hypothesis is true the expected influence is zero. Marks on the bars indicate the standard deviation of the influence of the sample under the null hypothesis. Finally the bars are coloured to distinguish the samples. In a logistic model the colours differentiate between the original groups, in an unadjusted linear model they differentiate the values above the mean from the values below the mean of Y. In an adjusted linear or the survival model they distinguish positive from negative residuals after fitting the null model.

Again, either of the commands below gives the same output.

```
> sampleplot(gt.cc)
> sampleplot(gt.kegg, "04110")
```

```
> sampleplot(gt.kegg["04110"])
```

The options of `sampleplot` and the resulting *gt.barplot* object are be handled in the same way as described under "geneplot". The difference is that the option *scale* defaults to `TRUE` in `sampleplot`.



4.4 Checkerboard plot

The fourth and fifth diagnostic plot can both also be used to assess the influence of each of the samples on the test result. The checkerboard plot visualizes the similarity between samples. It makes a square figure with the samples both on the X and on the Y-axis, so that it has all comparisons between the samples. Samples which are relatively similar are coded white and samples which are relatively dissimilar are coded black.

For easier interpretation the samples are sorted by their clinical outcome. If the test was (very) significant and the clinical outcome has two values, a typical block-like structure will appear. If the clinical outcome was continuous and the test is significant, the black squares will tend to stick together around the corners. By looking at these patterns some things can be learned about the structure of the data. For example, by looking at samples which deviate from the main pattern, outlying samples can be detected.

```
> checkerboard(gt.cc)
> checkerboard(gt.kegg, "04110")
```

The function `checkerboard` also has options *labelsize* and *drawlabels*. It returns a legend to link the numbers appearing in the plot if *drawlabels* = *FALSE* to the sample names.

4.5 Regression Plot

Using the regression plot an assessment can be made of the influence of each sample on the result of the test. It is an alternative visualization of the `sampleplot`.

The regression plot plots all pairs of samples, just like the checkerboard plot, but now showing the covariance between their clinical outcomes on the X-axis and the covariance between their gene expression patterns on the Y-axis. The comparisons of each sample with itself have been excluded.

The test statistic of the Global Test can be seen as a regression-coefficient for this plot, so it is visualized by drawing a least squares regression line. If this regression line is steep, the test statistic has a large value (and is possibly significant).

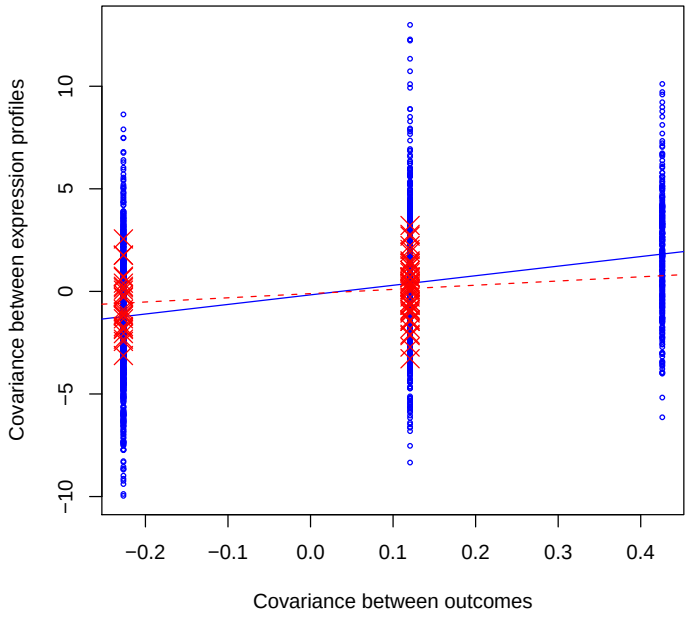
The influence of specific samples can be assessed by drawing a second regression line through only those points in the plot, which are comparisons involving the sample of interest. For example if we are interested the sample with sample name `"1"`, we take the points corresponding to the pairs (1,2) up to (1,72). If the regression line drawn through only these points deviates much from the general line, the sample deviates from the general pattern. This is especially the case if this line has a negative slope, which means that the sample is more similar (in its gene expression pattern) to the samples with a different clinical outcome than to samples with a similar clinical outcome.

If we want to test sample `"1"`, we say:

```
> regressionplot(gt.cc, sampleid = "1")
> regressionplot(gt.kegg, "04110", sampleid = "1")
```

We can also use this plot for a group of samples, saying for example:

```
> regressionplot(gt.cc, sampleid = c("1", "2"))
```



5 Reference

- J. J. Goeman, F. de Kort, S. A. van de Geer and J. C. van Houwelingen, 2004, 'A global test for groups of genes: testing association with a clinical outcome' *Bioinformatics* 20 (1) 93–99.
- J. J. Goeman, J. Oosting, A. Cleton-Jansen, J. K. Anninga and J. C. van Houwelingen, 2005, 'Testing association of a pathway with survival' *Bioinformatics* 21 (9) 1950-1957.