Testing association of a pathway with a clinical variable

Jelle Goeman Jan Oosting

Package *globaltest*.

version 4.10.0

May 12, 2008

Contents

Important Note

As of version 4.0.0 the method for calculating the asymptotic p-value has much improved. To use gamma approximation results of earlier versions (and of the Bioinformatics paper add the option `method = "gamma"` to the `globaltest` function (only for compatibility). See section 3.4 for more details.

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 response. 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 following papers (see page 25 for full references).

- Goeman et al. (2004) introduces the principle of global testing in microarray data analysis.
- Goeman et al. (2005) extends the approach to a survival time response and explains how to deal with covariates.
- Goeman et al. (2006) derives the mathematical properties of the test and proves its optimality under some conditions.

In recent 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.

The globaltest method is based on a prediction model for predicting a response variable from the gene expression measurements of a set of genes. The null hypothesis to be tested is that expression profile of the genes in the gene set is not associated with the response variable. A significant test result has three equivalent interpretations (see Goeman et al., 2004, 2005, for more details).

- If the test is significant, part of the variance of the response variable can be predicted from the gene expression measurements of the gene set.
- If the test is significant, the genes in the gene set are, on average, more associated with the response variable than would be expected under the null hypothesis. These associations may be both positive (upregulation) and negative (downregulation). Typically, a significant pathway is a mix op positively and negatively associated genes.
- If the test is significant, samples with similar values of the response variable tend to have relatively similar expression profiles over the genes in the gene set. Therefore, in an unsupervised cluster analysis, samples with the same value of the response variable have a tendency to turn up in the same clusters.

By grouping together genes before testing usually the number of tests will decrease and the severity of the correction for multiple testing will decrease. Genes can be grouped together into genesets in any meaningful way, for example based in function (KEGG, GO) or location (chromosome, cytoband). However, the choice of the gene sets may not be based in any way on the response variable that is used for testing.

2 Preparation

In the examples that follow we use two data sets. Both use the dataformat use BioConductor *ExpressionSet* input, which is the standard format for storing gene expression data in BioConductor. The *ExpressionSet* is the preferred input format for data in *globaletest*, but simple vector/matrix input is also possible. See `help(globaltest)`.

2.1 Leukemia Data

The first data set is the famous Leukemia data set by Golub et al. (1999), that is available from the *golubEsets* package on BioConductor. We load the data and normalize them using *vsn*. We reduce the data set in size in order to speed up the computations in this Vignette.

```
> library(golubEsets)
> library(vsn)
> data(Golub_Merge)
> Golub <- Golub_Merge[, 1:34]
> exprs(Golub) <- exprs(vsn2(Golub_Merge[, 1:34]))
```

This gives us a data set `Golub`, which is of the format *ExpressionSet*. It has 7,129 genes for 34 samples on Affymetrix hu6800 chips. We use *vsn* to normalize the data here, but any other normalization method may be used instead. Several phenotype variables are available with `Golub`, among them “ALL.AML”, the clinical variable that interests us, which separates the samples into Acute Lymphoblastic Leukemia and Acute Myelogenous Leukemia samples.

2.2 Breast Cancer Data

The second data set is a breast cancer data set from the Dutch Cancer Institute (Van de Vijver et al., 2002), which we have have included (by kind permission from the authors) in the *globaltest* package. To keep the package small we have reduced the data set to only 230 probes (those probes annotated to 64 pathways (all related to cell cycle) contained in our example annotation `annotation.vandeVijver`) and only 100 random patients.

```
> library(globaltest)
> data(vandeVijver)
```

This loads another *ExpressionSet*, called `vandeVijver`, which contains the data. The chip was done with Rosetta technology and the data are already normalized by Rosetta. The response of primary interest is the survival time, coded in variables “TIMEsurvival” (the survival time) and “EVENTdeath” (which indicates whether the observed time indicates a death or a censoring).

2.3 Annotation

To be able to use *globaltest* one must provide annotation to link the gene sets to be tested to the genes in the data.

The easiest way to do this is to use the metadata packages provided in BioConductor. These are provided for most commercially available chips and for many organisms. For example, to get GO and KEGG annotation for the Leukemia data set, we use the *hu6800.db* package.

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

This creates a list `kegg` of 197 pathways and a list `go` of 7810 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 (IEA). 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.

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

The vectors `KEGG.cellcycle` (103 probes) and `GO.cellcycle` (307 probes) contain the probes on the hu6800 chip that are annotated to the Cell Cycle pathway.

There is no metaData package for the Rosetta chips used in the Breast Cancer Data. Some annotation has been included in the *globaltest* package (see `help(annotation.vandeVijver)`), but we will not use it in this vignette. See the help files.

3 The Global Test

This section explains the options of the `globaltest` function, which is the main function of the package.

3.1 Testing a single pathway

Suppose we are interested in testing whether AML and ALL patients have different overall gene expression profiles. We can test this by saying

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

The first input X should be the object with the expression data (here the *ExpressionSet* object), the second input Y should give the response variable. Because the call above provides an *ExpressionSet* for X, we only need to specify the name of the repsonse variable in Y. Alternatively, the user can provide the whole vector in Y. In that case X only needs to provide the matrix of gene expression measurements (but is may still be an *ExpressionSet*).

The third function argument should be the gene set(s) to be tested. The default is to test the gene set of all genes on the chip.

The test result is stored in a *gt.result* object, which also contains all the information needed to draw the diagnostic plots. Many methods offer access to the information contained in the gt.result. Type `help(gt.result)` for an overview.

```
> gt.all

Global Test result:
Data: 34 samples with 7129 genes; 1 gene set
Model: logistic
Method: Asymptotic distribution

    Genes Tested Statistic Q Expected Q sd of Q    P-value
all  7129   7129      33.321         10  3.1264 2.9649e-05
```

This gives a summary of the test and its result. The model was "logistic" because the response `ALL.AML` is two-valued. The p-value was calculated using the asymptotic distribution (see section 3.4 for details). The bottom part is the test result. The gene set of all genes has 7,129 genes, which were all included in the test. Then it gives the test statistic Q, its expectation and standard deviation under the null hypothesis, and finally the p-value.

From this we conclude that there is ample evidence that the overall gene expression profile for all 7,129 genes is associated with the response. This means that samples with the same AML/ALL status tend to have similar expression profiles. It also means that the expression profile of a non-negligible proportion of the 7,129 genes is differentially expressed between AML and ALL patients. And it also means that there is potential for prediction of ALL/AML status from the gene expression profile over all 7,129 genes.

In cases such as this one, in which the overall expression pattern is (highly) associated with the response variable, we can expect many pathways (especially the larger ones) also to be associated with the response variable. See section 5 for a way to deal with this situation.

To test a specific pathway, provide it as the third argument *genesets* to `globaltest`.

```
> globaltest(Golub, "ALL.AML", GO.cellcycle)

Global Test result:
Data: 34 samples with 7129 genes; 1 gene set
Model: logistic
Method: Asymptotic distribution

     Genes Tested Statistic Q Expected Q sd of Q    P-value
[1,]   307    262      34.485     11.064  4.0189 0.00047335
```

Just as with the gene set of all genes, we conclude from the test result that the expression profile of the cell cycle pathway (as defined by GO) is notably different between AML and ALL samples. Patients with AML tend to be more similar to other AML patients than to ALL patients, with respect to their cell cycle expression profile (and vice versa).

Note that in the test result for the cell cycle pathway there is a difference between the values of "Genes" and "Tested". "Genes" just gives the length of the input vector that defined the gene set, while "Tested" gives the number of unique probe ids that can be matched to probe ids in the gene expression matrix. The vector `GO.cellcycle` is of length 307, but only contains 262 unique probe ids. Duplicate probe ids are automatically found and left out by `globaltest`.

3.2 Multiple Global Testing

It is possible to test many pathways at once by providing a *list* of pathways. For example:

```
> cellcycle <- list(go = GO.cellcycle, kegg = KEGG.cellcycle)
> globaltest(Golub, "ALL.AML", cellcycle)

Global Test result:
Data: 34 samples with 7129 genes; 2 gene sets
Model: logistic
Method: Asymptotic distribution

     Genes Tested Statistic Q Expected Q sd of Q    P-value
go     307    262      34.485    11.0640  4.0189 0.00047335
kegg   103    103      35.370     9.6216  3.9502 0.00038561
```

We have already made lists of all KEGG and GO pathways in the Section 2.3, and stored them as `kegg` and `go`. To test all KEGG pathways we say:

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

We will not display the whole result `gt.kegg`, but we can display part of it. To get the cell cycle result (KEGG 04110), or to get the first ten results, we can say

```
> gt.kegg["04110"]
> gt.kegg[1:10]
```

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

```
> sorted <- sort(gt.kegg)
> top5 <- sorted[1:5]
> top5

Global Test result:
Data: 34 samples with 7129 genes; 5 gene sets
Model: logistic
Method: Asymptotic distribution

      Genes Tested Statistic Q Expected Q sd of Q    P-value
```

```
04662    64     64      134.230     14.7970  8.0552 6.5298e-07
04610    81     81       75.576      9.8109  4.8200 1.1059e-06
00480    25     25       71.230     10.9770  5.5011 1.8584e-06
04640   118    118      105.290     16.0360  7.2968 2.0658e-06
00980    50     50       38.676      7.4761  3.1262 3.3022e-06
```

To make this list more readable, the pathway numbers can be replaced by the pathway names. For this we need the *KEGG.db* package, which contains the necessary information.

```
> library(KEGG.db)
> names(top5) <- as.list(KEGGPATHID2NAME)[names(top5)]
> top5
```

```
Global Test result:
Data: 34 samples with 7129 genes; 5 gene sets
Model: logistic
Method: Asymptotic distribution

                                             Genes Tested Statistic Q
B cell receptor signaling pathway               64     64     134.230
Complement and coagulation cascades             81     81      75.576
Glutathione metabolism                          25     25      71.230
Hematopoietic cell lineage                     118    118     105.290
Metabolism of xenobiotics by cytochrome P450    50     50      38.676
                                             Expected Q sd of Q    P-value
B cell receptor signaling pathway               14.7970  8.0552 6.5298e-07
Complement and coagulation cascades              9.8109  4.8200 1.1059e-06
Glutathione metabolism                          10.9770  5.5011 1.8584e-06
Hematopoietic cell lineage                      16.0360  7.2968 2.0658e-06
Metabolism of xenobiotics by cytochrome P450     7.4761  3.1262 3.3022e-06
```

There are some other functions to extract useful information from a *gt.result* object. See also `help(gt.result)`. The most important are:

```
> p.value(top5)
```

```
           B cell receptor signaling pathway
                                6.529794e-07
         Complement and coagulation cascades
                                1.105926e-06
                      Glutathione metabolism
                                1.858399e-06
                  Hematopoietic cell lineage
                                2.065801e-06
Metabolism of xenobiotics by cytochrome P450
                                3.302156e-06
```

```
> names(top5)
```

```
[1] "B cell receptor signaling pathway"
[2] "Complement and coagulation cascades"
```

```
[3] "Glutathione metabolism"
[4] "Hematopoietic cell lineage"
[5] "Metabolism of xenobiotics by cytochrome P450"
```

```
> result(top5)
```

```
                                              Genes Tested Statistic Q
B cell receptor signaling pathway                64     64   134.23070
Complement and coagulation cascades              81     81    75.57591
Glutathione metabolism                           25     25    71.22976
Hematopoietic cell lineage                      118    118   105.28606
Metabolism of xenobiotics by cytochrome P450     50     50    38.67566
                                              Expected Q  sd of Q      P-value
B cell receptor signaling pathway              14.796649 8.055237 6.529794e-07
Complement and coagulation cascades             9.810950 4.819992 1.105926e-06
Glutathione metabolism                         10.977495 5.501085 1.858399e-06
Hematopoietic cell lineage                     16.035759 7.296770 2.065801e-06
Metabolism of xenobiotics by cytochrome P450    7.476096 3.126186 3.302156e-06
```

```
> length(top5)
```

```
[1] 5
```

When testing more than one pathway, it is important to correct for multiple testing. The function `multtest` calculates multiplicity-adjusted p-values.

```
> sorted <- gt.multtest(sorted, "FWER")
> sorted[1:5]
```

```
Global Test result:
Data: 34 samples with 7129 genes; 5 gene sets
Model: logistic
Method: Asymptotic distribution

      Genes Tested Statistic Q Expected Q sd of Q    P-value FWER.adjusted
04662    64     64     134.230    14.7970  8.0552 6.5298e-07    0.00012864
04610    81     81      75.576     9.8109  4.8200 1.1059e-06    0.00021676
00480    25     25      71.230    10.9770  5.5011 1.8584e-06    0.00036239
04640   118    118     105.290    16.0360  7.2968 2.0658e-06    0.00040077
00980    50     50      38.676     7.4761  3.1262 3.3022e-06    0.00063732
```

The second option *proc* takes values *FWER* for the family-wise error rate (Holm's method), and *FDR* for the false dicovery rate (Benjamini and Hochberg). Be careful not to apply the `gt.multtest` function on a collection of pathways selected on their p-values. For more sophisticated multiple testing adjustments, see the *multtest* package, and Section 4.

3.3 Different types of response variable

The Global Test allows four different types of response variables to be tested. We give examples of each below. The statistical model that `globaltest` uses is usually determined from the input of the response variable, but the automatic choice can be overridden using the function argument *model*.

Two-valued response This is the most common type of response variable in microarray data analysis. If the response variable takes only two values, `globaltest` automatically chooses the logistic regression model. See the examples in the previous two sections.

A multi-valued response can be transformed to a two-valued response with the option *levels*. See Multi-valued Response, below.

Multi-valued response If the response is a *factor* which takes more than two values, `globaltest` automatically chooses a multinomial logistic regression model. For example, in the Leukemia data, we might want to know whether the overall expression profile depends on the center the samples came from (coded as `Source`) in `Golub`. We can test this with

```
> globaltest(Golub, "Source")

Global Test result:
Data: 34 samples with 7129 genes; 1 gene set
Model: multinomial
Method: Asymptotic distribution

    Genes Tested Statistic Q Expected Q sd of Q    P-value
all  7129   7129      23.125          10  1.9632 2.5099e-05
```

It might have been expected that the expression profiles are different for different centers, as many centers provided only ALL or only AML samples. We can compare the samples from only the centers CALGB and CCG (which both provided only AML samples) using the option *levels*:

```
> globaltest(Golub, "Source", levels = c("CALGB", "CCG"))

Global Test result:
Data: 9 samples with 7129 genes; 1 gene set
Model: logistic
Method: All 126 permutations

    Genes Tested Statistic Q Expected Q sd of Q P-value
all  7129   7129      12.746      11.25  1.8244 0.19048
```

For a more sophisticated way of correcting for confounders (such as `ALL.AML` in this case), see Section 3.5.

If only one level is given in *levels*, that outcome category is tested against the other categories combined.

```
> globaltest(Golub, "Source", levels = "St-Jude")

Global Test result:
Data: 34 samples with 7129 genes; 1 gene set
Model: logistic
Method: Asymptotic distribution

    Genes Tested Statistic Q Expected Q sd of Q   P-value
all  7129   7129      20.017          10  3.1264 0.0078509
```

Note that if a multi-valued response is not explicitly coded as *factor*, globaltest will see it as continuous. The easiest way to let globaltest know that the response should not be treated as continuous is to make it a factor. This can be done with

```
> globaltest(Golub, "factor(Source)")
```

This is especially relevant for time series, as the time points in a time series should usually be tested with a multinomial model, but they are not usually coded as *factor*.

Continuous response If the response is continuous, `globaltest` uses a linear model. For example, in the Leukemia data, the percentage of blast cells was measured for the 15 AML samples from CALGB. To test whether this percentage is reflected in the overall expression profile, we can say:

```
> calgb <- pData(Golub)["Source"] == "CALGB"
> globaltest(Golub[, calgb], "pctBlasts")

Global Test result:
Data: 4 samples with 7129 genes; 1 gene set
Model: linear
Method: All 24 permutations

    Genes Tested Statistic Q Expected Q sd of Q P-value
all  7129   7129      8.4321         10  2.4961   0.625
```

Survival as response The primary response in the Breast Cancer Data set is survival of patients. Survival time should be coded as a time for each patient (which is the last observation time for that patient) plus an *event indicator*, a dummy that indicates whether the patient died (had an event) at that point or was still alive (censored) at the last observation time.

```
> globaltest(vandeVijver, "Surv(TIMEsurvival, EVENTdeath)")

Global Test result:
Data: 100 samples with 230 genes; 1 gene set
Model: survival
Method: Asymptotic distribution

    Genes Tested Statistic Q Expected Q sd of Q   P-value
all   230    230       3.039          0       1 0.0011867
```

By convention, the event indicator takes a 1 if the patient died, and zero otherwise. If the event indicator is coded differently, the value that indicates an event should be explicitly given. The following alternative calls give the same output as the above:

```
> globaltest(vandeVijver, "Surv(TIMEsurvival, EVENTdeath == 1)")
> globaltest(vandeVijver, "TIMEsurvival", d = "EVENTdeath")
> globaltest(vandeVijver, "TIMEsurvival", d = "EVENTdeath", event = 1)
```

3.4 Different methods for calculating the p-value

The global test can calculate the p-value using different methods. The most important ones being permutations and the asymptotic distribution. The method to be used can be specified using the option *method*.

Asymptotic method This method calculates the p-value based on the asymptotic distribution. This is the recommended method for large sample sizes. It can be slightly conservative for small samples. This new asymptotic method supersedes the Gamma approximation described in Goeman et al. (2004).

For survival as response this uses the normal distribution as described in Goeman et al. (2005).

For continuous, two-valued and multi-valued response the asymptotic method uses numerical methods of Box (1954) and Kotz et al. (1967) for calculating the distribution of non-chi-squared distributed quadratic forms. The calculation of the p-values involves a step that smoothes away very small eigenvalues. An extra function argument *accuracy* can be given when calling globaltest to control the degree of smoothing. Low values of *accuracy* speed up the calculations but may result in somewhat conservative p-values. High values of *accuracy* result in slow calculations but accurate p-values. At the default value $accuracy = 50$ the conservativeness of the p-value is less than 0.1%. P-values below 10^{-12} are numerically inaccurate for the asymptotic method and are reported as zero.

Exact permutation method If the number of possible permutations of the response values of the samples is small, it is possible to list all permutations and calculate an exact permutation based p-value.

Random permutation method If the number of possible permutations of the response values of the samples is large, one can take a random sample from all possible permutations. This gives an approximate permutation based p-value. This permutation p-value is not so accurate in the lower range as it is always a multiple of one over the number of permutations used. It also has some sampling variation, so that two calls to `globaltest` using this method will usually not give exactly the same p-values.

Gamma method This method uses the gamma distribution (also referred to as *scaled chi-squared*) as an approximation to the asymptotic distribution. This has the advantage that it is slightly quicker to calculate and that it is less conservative for small sample sizes. A drawback is that it can be anti-conservative, especially for large gene sets. This is the distribution that was used in Goeman et al. (2004). We recommend using the new asymptotic method instead. See above. There is no gamma approximation if the response variable is a survival time.

The user chooses the method by specifying the arguments *method* and *nperm*. The default `method = "auto"` is to use the exact permutation method if the number of possible permutations is less than the option *nperm* (default 10,000) and to use the asymptotic method otherwise. The threshold of 10,000 permutations is reached around 17 samples for a two-valued response and at 8 samples for a continuous or survival response. A second option is `method = "permutation"`, which also chooses the exact permutation method if the number of possible permutations is less than *nperm* (or 10,000), but uses *nperm* random permutations otherwise. Alternatively, `method = "asymptotic"` and `method = "gamma"` always use the asymptotic and gamma methods, respectively. Prior to version 4.0.0 the default method was `method = "gamma"`.

An example using the permutations method:

```
> globaltest(Golub, "ALL.AML", GO.cellcycle, method = "p", nperm = 1000)

Global Test result:
Data: 34 samples with 7129 genes; 1 gene set
Model: logistic
Method: 1000 random permutations

     Genes Tested Statistic Q Expected Q sd of Q P-value
[1,]   307    262      34.485     11.433  2.8816       0
```

Note that the different methods also have different ways of calculating the mean and standard deviation of the test statistic Q.

There is also a function called `permutations`, which recalculates the p-value using the permutation method, and which can also be used to later increase the number of permutations.

```
> gt <- globaltest(Golub, "ALL.AML", GO.cellcycle)
> permutations(gt, nperm = 2000)

Global Test result:
Data: 34 samples with 7129 genes; 1 gene set
Model: logistic
Method: 2000 random permutations

     Genes Tested Statistic Q Expected Q sd of Q P-value
[1,]   307    262      34.485     11.433  2.9409       0
```

All permutation test statistics are stored in the *gt.result* object for later use (for example in the function `hist`, see below). Using the permutation method on many pathways may therefore lead to memory problems.

3.5 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 Leukemia Data we may be concerned about a possible 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 by incorporating the covariate `BM.PB`, which codes for bone marrow or peripheral blood, into the null hypothesis.

If covariates are incorporated into the null hypothesis of the Global Test, it tests whether the gene expression profile has an independent association with the response that cannot be explained away by the presence of the confounding covariates. For the statistical details of the model, see Goeman et al. (2005).

The three interpretations of the resulting test are now as follows

- If the test is significant, part of the *residual* variance of the response variable, *that could not be predicted from the included covariates*, can be predicted from the gene expression measurements of the gene set.

- If the test is significant, the genes in the gene set are, on average, more associated with the *residual* response variable than expected under the null hypothesis. These associations may be both positive (upregulation) and negative (downregulation). Typically, a significant pathway is a mix of positively and negatively associated genes. *The associations between the genes and the response are therefore genuine and cannot be explained by confounding with the included covariates.*

- If the test is significant, samples with similar values of the *residual* response variable tend to have relatively similar expression profiles over the genes in the gene set.

The Global Test will generally lose power if the covariate that is adjusted for is highly associated with the response variable. The test may also gain power if the confounding covariate is not associated with the response, but is a source of variation in the gene expression data.

The easiest way to adjust for confounders is to use a *formula* object. For example, in the Leukemia Data we can adjust for the confounder `BM.PB` with (note the absence of quotes!)

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

Global Test result:
Data: 34 samples with 7129 genes; 1 gene set
Model: logistic, ALL.AML ~ BM.PB
Adjusted: 100 % of variance of Y remains after adjustment
Method: Asymptotic distribution

     Genes Tested Statistic Q Expected Q sd of Q    P-value
[1,]   307    262       36.29     10.919  3.9847 0.00025611
```

There is still proof of a large difference in the cell cycle gene expression profile between AML and ALL patients if we correct for the difference in the way the samples were obtained.

In the linear, logistic and multinomial models `globaltest` will also give a percentage of the variance of the response variable that is lost in the adjustment. This gives an indication of the amount of variation in the response variable that the confounder explains and of the power lost in the adjustment process.

To adjust for more than one covariate or for interaction effects, one can use all possibilities offered by the *formula* object. See `help(formula)` for more details. In the Breast Cancer Data we can test to see if the gene expression profiles have anything to add to the prediction of survival from the known risk indicators included in the data: `NIH` and `ESR1` (oestrogen receptor status), and `Posnodes` (lymph node status):

```
> globaltest(vandeVijver, Surv(TIMEsurvival, EVENTdeath) ~ NIH +
      ESR1 + Posnodes)
```

```
Global Test result:
Data: 100 samples with 230 genes; 1 gene set
Model: survival, Surv(TIMEsurvival, EVENTdeath) ~ NIH + ESR1 + Posnodes
Method: Asymptotic distribution

    Genes Tested Statistic Q Expected Q sd of Q  P-value
all   230    230      1.2906          0       1 0.098423
```

From the result we see that the microarray data do not seem to have any additional predictive value in this reduced data set. The p-value rose substantially relative to the unadjusted test, so at least one of these known risk factors is expected to be a confounder: it is strongly associated with survival and also with the gene expression data. This latter statement can be checked by using the covariate as the response

```
> globaltest(vandeVijver, "ESR1")

Global Test result:
Data: 100 samples with 230 genes; 1 gene set
Model: logistic
Method: Asymptotic distribution

    Genes Tested Statistic Q Expected Q sd of Q P-value
all   230    230       126.5         10  3.9274       0
```

This shows that `ESR1` is clearly associated with gene expression.

The model that was fitted under the null hypothesis of the test can be retrieved and displayed with the command `fit`.

```
> gt <- globaltest(vandeVijver, Surv(TIMEsurvival, EVENTdeath) ~
      ESR1)
> fit(gt)

Call:
coxph(formula = ff, data = pData, na.action = "na.fail", x = TRUE,
    y = TRUE)


      coef exp(coef) se(coef)     z      p
ESR1 -1.15     0.316    0.398 -2.90 0.0038

Likelihood ratio test=7.56  on 1 df, p=0.00597  n= 100
```

This shows that oestrogen receptor status is also clearly associated with survival. Try `summary(fit(gt))` for a more detailed output.

4 Multiple testing on the GO graph

When testing Gene Ontology terms with the *globaltest* package, it is possible to make a more sophisticated multiple testing adjustment than the one offered by the `gt.multtest` function.

The *focus level procedure* implemented in the function `gtGO` is a multiple testing procedure that controls the family-wise error rate while preserving the structure of the GO graph. It returns a significant subgraph of the GO graph in which the probability of any error in that graph is controlled at level α.

First, the GO graph must be prepared for the data set at hand.

```
> bp <- makeGOstructure(Golub, "hu6800", unreliable = "IEA")
> bp
```

```
A GO structure object for globaltest with 4530 GO terms.
```

This created a *GOstructure* object for the Golub data, using the annotation package *hu6800*. It has 4530 GO terms. By default the Biological Process subgraph of GO is used. This can be changed with by setting *ontology = "CC"* or *ontology = "MF"*. The *unreliable* argument can be used, as above, to specify that some evidence codes should be considered unreliable. Further, the graph can be made smaller by specifying a different top node in *top*, in which case only the descendants of that top node are used. Alternatively, by specifying *only.nodes* the GO structure is restricted to only the nodes specified.

The focus level procedure requires the specification of a *focus level*. This is the level in the GO graph at which the focus level starts looking for significant GO terms. Only after finding significant terms at the focus level will the procedure expand its scope to higher or lower level terms. The focus level can therefore be used to determine the level of specificity of the GO terms that are of most interest. A focus level can be specified as any vector if GO identifiers, but a useful automated way to make the focus level is by using the function `getFocus`, which ensures a certain level of generality, while keeping computation time limited.

```
> focusBP <- getFocus(bp, maxatoms = 7)
```

The `getFocus` function accepts an extra argument *maxatoms* (default: 10), which determines the maximum complexity of the subgraphs of all descendants of focus level nodes. Lower values of *maxatoms* lead to more specific GO terms in the focus level and quicker calculations, while higher values lead to more general GO terms in the focus level and slower calculations.

The significant subgraph of size 60 can be calculated with

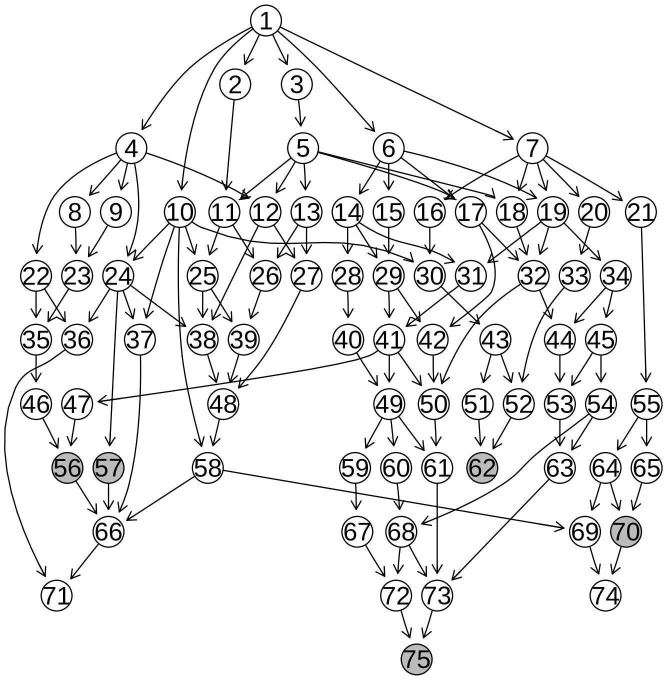
```
> go60 <- gtGO(Golub, "ALL.AML", focus = focusBP, GO = bp, stopafter = 60)
```

Any arguments that can be given to `globaltest` can also be given to `gtGO` to specify the test that should be done on each GO term. The arguments *stopafter* and *maxalpha* govern when to stop the algorithm: it stops either when a certain number of significant GO terms is found, or when certain alpha-level is reached.

To visualize the result we can use the *GOstats* and *Rgraphviz* packages, which offer many possibilities. For example, we can plot the significant GO graph with numbers and a legend and colour in grey all nodes corresponding to the focus level.

```
> library(GOstats)
> library(Rgraphviz)
> sigGO <- GOGraph(names(go60), GOBPPARENTS)
> sigGO <- removeNode("all", sigGO)
> sigGO <- as(t(as(sigGO, "matrix")), "graphNEL")
> nodes <- buildNodeList(sigGO)
> focusnode <- sapply(nodes, name) %in% focusBP
> names(focusnode) <- names(nodes)
> nodefill <- ifelse(focusnode, "#BBBBBB", "white")
> nAttrs <- list()
> nAttrs$fillcolor <- nodefill
> nAttrs$label <- 1:length(names(nodes))
> names(nAttrs$label) <- names(nodes)
> pg <- plot(sigGO, nodeAttrs = nAttrs)
> x <- getNodeXY(pg)$x
> y <- getNodeXY(pg)$y
> ordering <- sort.list(order(-y, x))
> nAttrs$label <- ordering
> names(nAttrs$label) <- names(nodes)
> plot(sigGO, nodeAttrs = nAttrs)
> Terms <- sapply(lookUp(names(nodes)[sort.list(ordering)], "GO",
     "TERM"), Term)
> names(Terms) <- NULL
> legend <- data.frame(Terms)
```

Figure 1: A significant GO subgraph for the Golub data



```
> legend
                                                                Terms
1                                                  biological_process
2                                    multicellular organismal process
3                                               biological regulation
4                                                response to stimulus
5                                    regulation of biological process
6                                                   metabolic process
7                                                    cellular process
8                                                  response to stress
9                                       response to external stimulus
10                                              immune system process
11                     regulation of multicellular organismal process
12                                 regulation of response to stimulus
13                          positive regulation of biological process
14                                    macromolecule metabolic process
15                                          primary metabolic process
16                                                    cell activation
17                                    regulation of metabolic process
18                                     regulation of cellular process
19                                         cellular metabolic process
20                                                 cell proliferation
21                                                 cell communication
22                                                   defense response
23                                               response to wounding
24                                                    immune response
25                                regulation of immune system process
26           positive regulation of multicellular organismal process
27                        positive regulation of response to stimulus
28                                       biopolymer metabolic process
29                                          protein metabolic process
30                                               leukocyte activation
31                           cellular macromolecule metabolic process
32                           regulation of cellular metabolic process
33                                     mononuclear cell proliferation
34                                       phosphorus metabolic process
35                                              inflammatory response
36                                             innate immune response
37                                            immune effector process
38                                      regulation of immune response
39                       positive regulation of immune system process
40                                            biopolymer modification
41                                 cellular protein metabolic process
42                            regulation of protein metabolic process
43                                              lymphocyte activation
44                         regulation of phosphorus metabolic process
45                                        phosphate metabolic process
46                                        acute inflammatory response
47                                                        proteolysis
48                             positive regulation of immune response
49                                       protein modification process
50                   regulation of cellular protein metabolic process
51                                                  B cell activation
52                                           lymphocyte proliferation
53                          regulation of phosphate metabolic process
54                                                    phosphorylation
55                                                signal transduction
56   activation of plasma proteins during acute inflammatory response
57                                            humoral immune response
58                                      activation of immune response
59                                   peptidyl-amino acid modification
60                            post-translational protein modification
61                         regulation of protein modification process
62                                               B cell proliferation
63                                      regulation of phosphorylation
64                     immune response-regulating signal transduction
65                   cell surface receptor linked signal transduction
66                                              complement activation
67                                     peptidyl-tyrosine modification
68                                 protein amino acid phosphorylation
69                     immune response-activating signal transduction
70 immune response-regulating cell surface receptor signaling pathway
71                         complement activation, alternative pathway
72                                  peptidyl-tyrosine phosphorylation
73                   regulation of protein amino acid phosphorylation
74 immune response-activating cell surface receptor signaling pathway
75                    regulation of peptidyl-tyrosine phosphorylation
```

The plot legend can be made interactive using the following code

```
> repeat {
      p <- locator(n = 1)
      if (is.null(p))
          break()
      pg <- plot(sigGO, nodeAttrs = nAttrs)
      x <- getNodeXY(pg)$x
      y <- getNodeXY(pg)$y
      distance <- abs(p$x - x) + abs(p$y - y)
      idx <- which.min(distance)
      legend("topleft", legend = c(nAttrs$label[idx], names(focusnode)[idx],
          Term(lookUp(names(focusnode)[idx], "GO", "TERM")[[1]])),
          bg = "white")
  }
```

Clicking on any node now produces a legend for that node in the top left corner of the graph. Press escape to return to the `R` prompt.

5 The Comparative P

In some data sets, such as the two example data sets studied in this vignette, the Global Test for all genes is very significant. In this situation we can expect a substantial number of genes to be associated with the response variable. As a consequence, we can also expect many gene sets, especially the larger ones, also to be associated with the response.

A useful diagnostic to see whether a gene set is exceptionally significant is the "comparative p", which can be calculated using the function `sampling`.

```
> gt <- globaltest(Golub, "ALL.AML", KEGG.cellcycle)
> sampled.gt <- sampling(gt)
> sampled.gt
```

```
Global Test result:
Data: 34 samples with 7129 genes; 1 gene set
Model: logistic
Method: Asymptotic distribution

     Genes Tested Statistic Q Expected Q sd of Q    P-value Comparative p
[1,]   103    103       35.37     9.6216  3.9502 0.00038561         0.339
```

This gives an extra output column "Comparative p", which is the fraction of random genesets of the same size as the cell cycle pathway (103 genes) which have a larger standardized test statistic than the cell cycle pathway itself. Pathways of the same size with a larger standardized test statistic will almost invariably also have a lower p-value. In this case around 34 % of 1,000 random 'pathways' of size 103 have a larger standardized test statistic than the cell cycle pathway. This indicates that, although the cell cycle pathway is clearly differentially expressed between AML and ALL samples (low p-value), it is not exceptionally significant for a pathway of its size in this dataset.

Just like the p-value based on random permutations, the comparative p is a random quantity that has some sampling variation. It is always a multiple of one over the number of random pathways drawn; the accuracy can be increased by increasing that number. By default 1,000 random sets are sampled; this number can be changed with the option *ndraws*.

The Comparative P is a very useful diagnostic. However, it is not a p-value in the classical statistical sense, because it is based on permuting genes, not biological samples. It should always be interpreted together with the regular p-value.

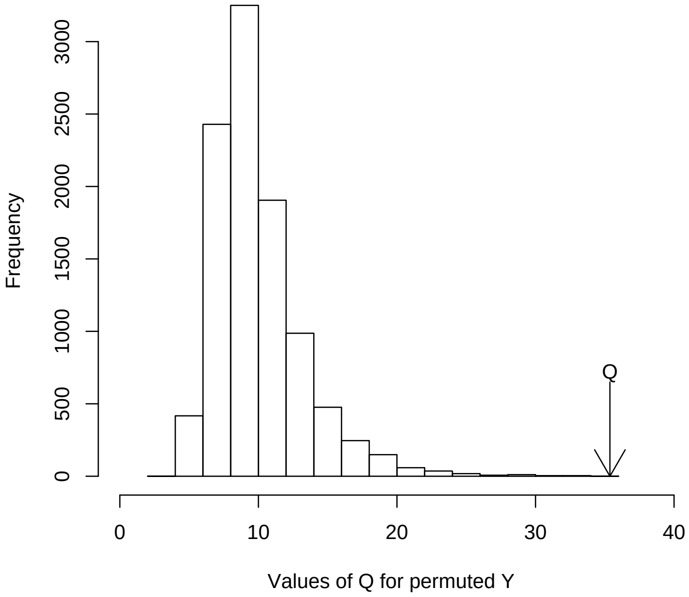
6 Diagnostic plots

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.

6.1 Permutations histogram

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

```
> gt <- globaltest(Golub, "ALL.AML", KEGG.cellcycle, method = "p")
> hist(gt)
```



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.

6.2 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 response (except in a survival model, where the expected height is zero). Marks indicate with how many standard deviations (under the null hypothesis) the bar exceeds the reference line. Finally the bars are colored to indicate a positive or a negative association of the gene with the response.

The geneplot influence bars have two interpretations. In the first place, each bar is 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 influence (relative to the reference line) or a negative influence from the pathway will generally result in a lower p-value for the pathway, removing a gene with a large positive influence will have the opposite effect.

```
> gt <- globaltest(Golub, "ALL.AML", kegg)
> geneplot(gt, "00561")
```

The second argument of `geneplot` is the name of the gene set to be plotted. This can be left out if `gt` contains only one gene set. 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. Similarly, the legend can be suppressed with *addlegend = FALSE*. Alternatively, one can store the geneplot as a *gt.barplot* object to retrieve the numbers or to plot (part of) the plot later, optionally with the option *plot = FALSE* to suppress plotting at this point.

```
> myplot <- geneplot(gt, "00561", plot = FALSE)
```

The return of the `geneplot` is an object of type *gt.barplot* containing the numbers and names appearing in the plot. This allows the user to customize the plot to his or her liking. For example, to increase interpretability, the probe identifiers appearing in the plot can be replaced by the gene symbols in this object and plotted again:

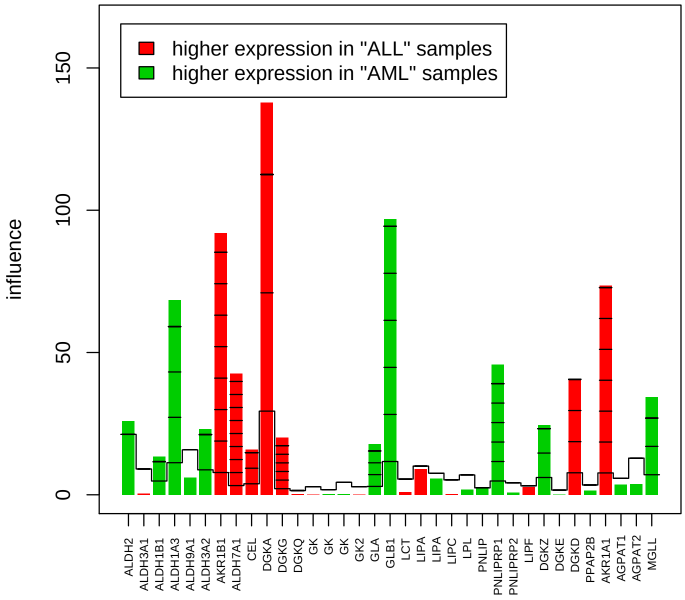
```
> names(myplot) <- as.list(hu6800SYMBOL)[names(myplot)]
> plot(myplot)
```

It is possible to supply new values for *genesubset*, *labelsize*, *drawlabels* or *addlegend* when plotting a *gt.barplot* object.

The *gt.barplot* object also allows one to look at subsets of a large pathway more closely. The gene plot can for example be sorted with the largest z-scores first. For example

```
> mysorted <- sort(myplot)
> top5 <- mysorted[1:5]
```

Figure 2: Example geneplot



The numbers can be retrieved with the following functions. `z.score` counts the number of standard deviations of the influence above the reference line. See `help(gt.barplot)` for more details.

```
> z.score(top5)

 ALDH7A1   AKR1B1   AKR1A1     DGKG PNLIPRP1
8.615686 7.614948 6.060994 5.970493 5.968965

> names(top5)

[1] "ALDH7A1"  "AKR1B1"   "AKR1A1"   "DGKG"     "PNLIPRP1"

> result(top5)

      name Influence Expected        SD  z.score     colouring
1  ALDH7A1  42.63253 3.233554  4.572936 8.615686 high in "ALL"
2   AKR1B1  92.03227 7.819780 11.058839 7.614948 high in "ALL"
3   AKR1A1  73.47796 7.676712 10.856510 6.060994 high in "ALL"
4     DGKG  20.19061 2.138031  3.023632 5.970493 high in "ALL"
5 PNLIPRP1  45.70046 4.840437  6.845412 5.968965 high in "AML"

> length(top5)

[1] 5
```

The option *scale* of `geneplot` can be used to rescale the bars to have unit standard deviation (so that the z-scores are displayed). Alternatively, this can be done later with the command `scale`.

6.3 Sample Plot

The Sample Plot looks very similar to the Gene Plot. It 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.

```
> gt <- globaltest(Golub, "ALL.AML", KEGG.cellcycle)
> sampleplot(gt)
```

The `sampleplot` result can be stored in a *gt.barplot* object just as with `geneplot`. The function arguments and the handling of the *gt.barplot* object are the same as for "geneplot" above. One important difference is that the option *scale* defaults to `TRUE` in `sampleplot`.

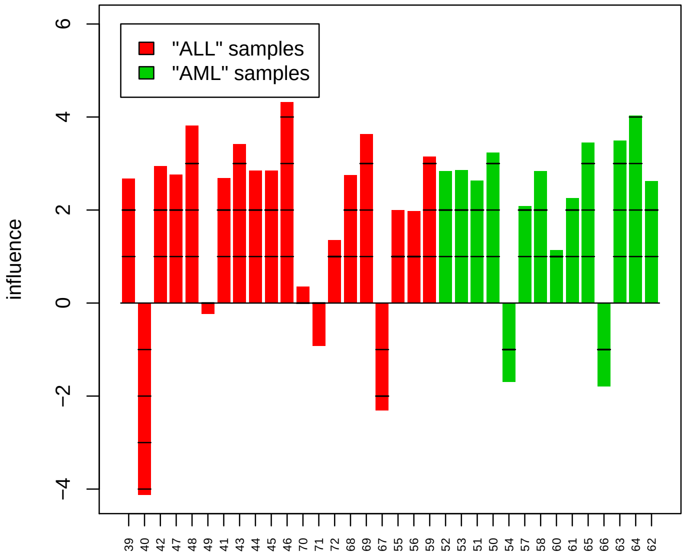
6.4 Other plots

The checkerboard and regression plots are two alternative plots to assess the influence of each of the samples on the test result.

Checkerboard plot 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 response value. If the test was (very) significant and the response has two values, a typical block-like structure will appear. If the response 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

Figure 3: Example sampleplot



example, by looking at samples which deviate from the main pattern, outlying samples can be detected.

```
> gt <- globaltest(Golub, "ALL.AML", kegg)
> checkerboard(gt.kegg, "04110")
```

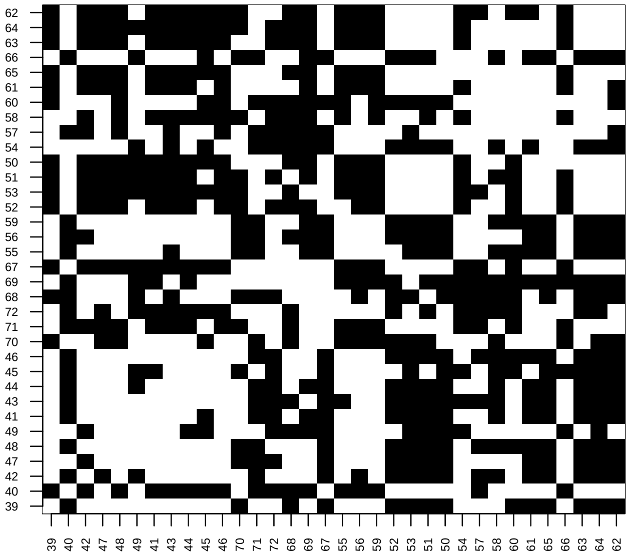
The function `checkerboard` also has options *labelsize* and *drawlabels*. It returns a legend to link the numbers appearing in the plot if $drawlabels = FALSE$ to the sample names.

Regression Plot The regression plot plots all pairs of samples, just like the checkerboard plot, but now showing the covariance between their response values 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 (1,34). 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

Figure 4: Example checkerboard plot



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 response than to samples with a similar response.

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

```
> gt <- globaltest(Golub, "ALL.AML", kegg)
> regressionplot(gt, "04110", sampleid = "39")
```

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

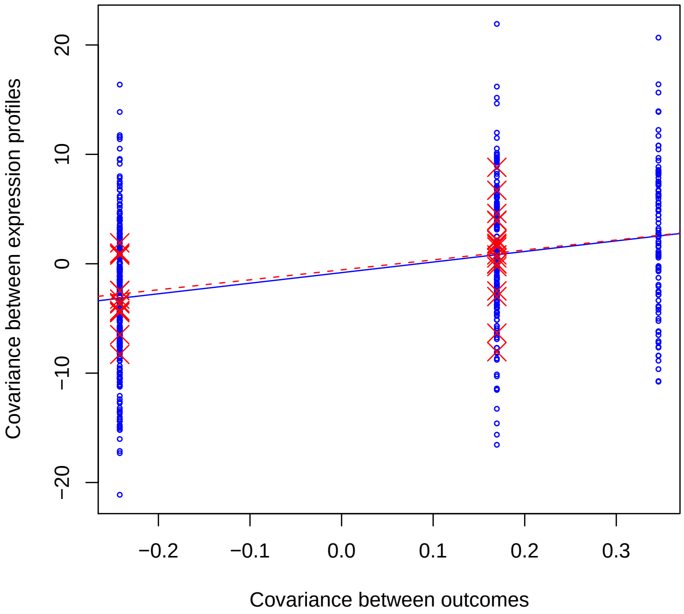
```
> regressionplot(gt, "04110", sampleid = c("39", "40"))
```

7 Session Information

The version number of R and packages loaded for generating the vignette were:

- R version 2.7.0 (2008-04-22), `i386-pc-mingw32`
- Locale: `LC_COLLATE=English_United States.1252;LC_CTYPE=English_United States.1252;LC_M`
- Base packages: base, datasets, graphics, grDevices, grid, methods, splines, stats, tools, utils

Figure 5: Example regressionplot



- Other packages: affy 1.18.0, affyio 1.8.0, annotate 1.18.0, AnnotationDbi 1.2.0, Biobase 2.0.1, Category 2.6.0, DBI 0.2-4, genefilter 1.20.0, globaltest 4.10.0, GO.db 2.2.0, golubEsets 1.4.4, GOstats 2.6.0, graph 1.18.1, hu6800.db 2.2.0, KEGG.db 2.2.0, lattice 0.17-7, limma 2.14.1, multtest 1.20.0, preprocessCore 1.2.0, RBGL 1.16.0, Rgraphviz 1.18.0, RSQLite 0.6-8, survival 2.34-1, vsn 3.6.0, xtable 1.5-2
- Loaded via a namespace (and not attached): cluster 1.11.10

References

Box, G. E. P. (1954). Some theorems on quadratic forms applied in the study of analysis of variance problems, I. Effect of inequality of variance in the one-way classification. *Annals of Mathematical Statistics*, 25:290–302.

Goeman, J. J., Oosting, J., Cleton-Jansen, A. M., Anninga, J. K., and van Houwelingen, J. C. (2005). Testing association of a pathway with survival using gene expression data. *Bioinformatics*, 21(9):1950–1957.

Goeman, J. J., van de Geer, S. A., de Kort, F., and van Houwelingen, J. C. (2004). A global test for groups of genes: testing association with a clinical outcome. *Bioinformatics*, 20(1):93–99.

Goeman, J. J., van de Geer, S. A., and van Houwelingen, J. C. (2006). Testing against a high-dimensional alternative. *Journal of the Royal Statistical Society Series B-Statistical Methodology*, 68(3):477–493.

Golub, T. R., Slonim, D. K., Tamayo, P., Huard, C., Gaasenbeek, M., Mesirov, J. P., Coller, H., Loh, M. L., Downing, J. R., Caligiuri, M. A., Bloomfield, C. D., and Lander, E. S. (1999). Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. *Science*, 286(5439):531–537.

Kotz, S., Johnson, N. L., and Boyd, D. W. (1967). Series representations of distributions of quadratic forms in normal variables. I. Central case. *Annals of Mathematical Statistics*, 38:823–837.

Van de Vijver, M. J., He, Y. D., van 't Veer, L. J., Dai, H., Hart, A. A. M., Voskuil, D. W., Schreiber, G. J., Peterse, J. L., Roberts, C., Marton, M. J., Parrish, M., Atsma, D., Witteveen, A., Glas, A., Delahaye, L., van der Velde, T., Bartelink, H., Rodenhuis, S., Rutgers, E. T., Friend, S. H., and Bernards, R. (2002). A gene-expression signature as a predictor of survival in breast cancer. *New England Journal of Medicine*, 347(25):1999–2009.