Significance Analysis of Function and Expression

William T. Barry
bill.barry@duke.edu

December 16, 2010

1 Introduction

This vignette demonstrates the utility and flexibility of the R-package `safe` in conducting tests of functional categories for gene expression studies. SAFE is a resampling-based method of testing that is applicable to many different experimental designs and sets of functional categories. SAFE extends and builds on an approach employed in Virtaneva *et al.* (2001), and defined more rigorously in recent publications from Barry *et al.* (2005 and 2008). It is suggested that all users refer to these publications in order to understand the SAFE terminology and principles in greater detail. We also ask that Barry *et al.* (2008) be cited in publications that use the updated version of `safe`.

Several of the extensions to the `safe` package are itemized below, and relate to added functionality discussed in Barry *et al.* (2008). Further, more functions and arguments are provided which improve the input and output capabilities of the package. Manuscripts for the citations mentioned above, and additional tutorials and examples are available at the following URL.

`http://www.duke.edu/~dinbarry/SAFE/`

2 Changes in version 2.0

The following list describes the extended capability of `safe` version 2.0. Examples of their implementation and the changes to functional arguments are illustrated in subsequent sections and detailed in help documents.

- Gene categories can be automatically generated within `safe` using the `platform` and `annotate` arguments. This can build categories from GO ontologies, KEGG pathways or PFAM domain, if a suitable Bioconductor annotation package exists for the array type.
- The sparse matrix package `SparseM` is incorporated to ease the memory constraints in working with large datasets and/or many hundreds of categories.
- Local statistics are added for analyses of paired data (`local = "t.paired"`), and a Cox proportional hazard model for censored survival data (`local = "z.COXPH"`).
- Global statistics are added for doing resampling-based tests of genelist-type analyses (`global = "Fisher"`) or (`global = "Pearson"`) or mean difference (`global = "AveDiff"`). See Barry et al. (2008) for further explanation of these global statistics.

- Non-resampling based error estimates are available including a Bonferroni correction or Holm's step-down procedure for the family-wise error rate (`error = "FWER.Bonf"`) or (`error = "FWER.Holm"`), and the Benjamini-Hochberg step-up procedure to control the false discovery rate (`error = "FDR.BH"`).
- The gene-specific results within a given category can be displayed using `gene.results`.
- SAFE results can be plotted across the directed graph of Gene Ontology using `safedag`.
- A bootstrap-based test is included which we have shown to be more powerful, require fewer resamples, and extendable to more complicated experimental designs with covariate information; see Barry *et al.* (2008) for more discussion of bootstrap-based tests.

3 SAFE implementation and output

Here, we implement `safe` using the datasets and annotations in Bioconductor packages listed below.

```
> library(safe)
> library(multtest)
> library(hu6800.db)
```

Every SAFE analysis requires three elements from a dataset: (1) gene expression data, (2) a response vector associated with the samples, and (3) a matrix containing category assignments that is either user-defined or built from annotation packages for the array platform.

The expression data should be in the form of an $m \times n$ matrix, where appropriate normalization and other pre-processing steps have been taken. It should be noted that in the current version of `safe`, missing values are not allowed in the expression data, and must be imputed prior to analysis. In this vignette, we will use the AML/ALL dataset from Golub *et al.* (1999) as illustration.

```
> data(golub)
> dimnames(golub)[[1]] <- golub.gnames[, 3]
```

`golub` is a matrix of normalized expression estimates for 3,051 genes across 38 samples. Row-names of Affymetrix hu6800 probeset IDs are added to `golub`. The row names are necessary for building gene categories on the subset of probesets retained in `golub`. The comparison of interest is between AML and ALL tumors subtypes. Tumor classification of samples is provided in `golub.cl` (AML = 1, ALL = 0). Section 4 will discuss the valid forms of response vectors for the experimental designs allowed in `safe`.

```
> table(golub.cl)

golub.cl
 0  1
27 11
```

For the primary example in this vignette, the functional categories of interest will be KEGG pathways. Pathway annotation for the Affymetrix array is available from the `hu6800` package. For the sake of parsimony, we will only consider pathways that have at minimum of 10 probeset members among the 3,051 in the `golub` dataset.

In version 2.0, the KEGG categories can now be automatically generated by the `safe` function, and is discussed is more detail in section 8.

NOTE: to be more efficient while running multiple examples, we will also generate a matrix of indicator variables for KEGG category membership externally from `safe` for repeated use. When working with user-defined category matrices, it is strongly suggested that appropriate names are given to all objects so the rows in the data and C.matrix correspond, and the output from `safe` is properly labeled.

```
> C.mat <- getCmatrix(gene.list = as.list(hu6800PATH), as.matrix = TRUE,
+     present.genes = golub.gnames[, 3], min.size = 10)

  Categories completed:20% 40% 60% 80% 100%
  142 catgories formed

> dimnames(C.mat)[[2]] <- paste("KEGG:", dimnames(C.mat)[[2]],
+     sep = "")

> set.seed(12345)
> results <- safe(golub, golub.cl, platform = "hu6800", annotate = "KEGG",
+     min.size = 10)
```

The SAFE framework for testing gene categories is a two-stage process, where "local" statistics assess the association between expression and the response of interest in a gene-by-gene manner, and a "global" statistic measures the extent of association in genes assigned to a category relative to their complement. As indicated, the default local statistic for the 2-sample comparison of AML and ALL is the Student's t-statistic. An increased amount of differential expression within a KEGG pathway is determined using a global Wilcoxon rank sum statistic by default. Inference on each type of statistic is achieved through permutation.

```
> results

SAFE results:
  Local: t.Student
  Global: Wilcoxon
  Method: permutation

           Size Mean.Rank Emp.pvalue
KEGG:00860   15    2412.2      0.004
KEGG:04110   57    1955.9      0.006
KEGG:04966   10    2312.9      0.015
KEGG:00561   12    2086.6      0.016
KEGG:05146   49    1821.9      0.016
KEGG:00240   30    1913.4      0.018
KEGG:03050   23    2108.7      0.021
KEGG:00970   16    2133.1      0.022
KEGG:05144   32    1953.0      0.024
KEGG:04920   27    1895.1       0.03
```

The basic output from `safe` is an object of class *SAFE*. Showing objects of class *SAFE* will print details on the type of analysis and the results for categories that attain a certain level of significance. Here, significant results are printed for the 6 categories that have empirical p-values ≤ 0.05. For each category, the number of annotated genes in the dataset is displayed along with the Wilcoxon global statistic and its empirical p-value. NOTE: as in standard gene-by-gene analyses, it is of critical importance to account for multiple comparisons when considering a number of categories simultaneously. Several options for adjusted p-values are provided in `safe`, and discussed in detail in Section 6.

Gene-specific results within a category are now made more readily accessible through the `gene.results` function. We believe this is very useful for investigators interested in seeing which category members are contributing to its significance. The following example demonstrates how the direction and magnitude of differential expression are displayed by default. A list of two data.frames can also be returned with the argument `print.it = FALSE`.

```
> gene.results(results, cat.name = "KEGG:00860")

Category gene-specific results:
  Local: t.Student
  Method: permutation

KEGG:00860 consists of 15 genes

Upregulated Genes
-----------------
               Local.Stat Emp.pvalue
D26308_at           5.498      0.001
M14016_at           4.338      0.001
X06985_at           4.206      0.001
D00726_at           3.800      0.001
M11147_at           3.718      0.002
L20941_at           3.706      0.002
Y00451_s_at         3.837      0.003
Z83821_cds2_at      2.856      0.003
M60891_s_at         2.705      0.018
M95623_cds1_at      2.372      0.022
M15182_at           1.781      0.072
U34877_at           1.717      0.094
X89267_at           0.663      0.563

Downregulated Genes
-------------------
               Local.Stat Emp.pvalue
X54326_at          -4.322      0.001
U82010_rna1_at     -2.125      0.041
```

4 Experimental Designs and Local Statistics

The basic 2-sample comparison in the example above is one of several experimental designs that `safe` can automatically accommodate. The following examples illustrate the arguments needed for the variety of designs and statistics allowed. In addition to the internal local statistics for generic comparisons, one can also employ user-defined functions in `safe` to extend its utility.

For 2-sample comparisons, the response vector can either be given as a (0,1) vector, or a character vector with two unique elements. It is important to note that when a character vector is passed to `safe` as the response, the assignment of the first array becomes Group 1, and is printed as a warning. Thus, the sign of the t-statistics have flipped below. NOTE: to decrease computation time, the permutation testing is bypassed by using the argument `Pi.mat = 1`.

By default, a Student's t-statistic is employed for 2-sample comparisons, but if unequal variances are assumed, the Welch t-statistic can be selected using `local="t.Welch"`.

```
> y.vec <- c("ALL","AML")[1+golub.cl]
> results2 <- safe(golub, y.vec, C.mat, Pi.mat = 1)

Warning: y.vec is not (0,1), thus Group 1 == ALL

> results3 <- safe(golub, golub.cl, C.mat, local="t.Welch", Pi.mat = 1)
> round(cbind(Student1 = results@local.stat[1:3],
+             Student2 = results2@local.stat[1:3],
+             Welch = results3@local.stat[1:3]),3)

                            Student1 Student2  Welch
AFFX-HUMISGF3A/M97935_MA_at    2.502   -2.502  1.759
AFFX-HUMISGF3A/M97935_MB_at    1.156   -1.156  0.910
AFFX-HUMISGF3A/M97935_3_at    -0.110    0.110 -0.098
```

For multi-class designs, response vectors should be character or numeric vectors with unique values for each group. If a character vector is supplied for `y.vec`, an ANOVA F-statistic is computed by default; otherwise, an ANOVA test can be specified with the argument `local = "f.ANOVA"` for numeric class assignments. Simple linear regression is performed if a numeric vector with more than two unique values is supplied, or by using the argument `local = "t.LM"`.

Version 2.0 of `safe` is extended to include the paired t-test for matched experiments. For this, samples are identified by (+/-) pairs of integers. Internally, the permutation algorithm changes from random sampling without replacement, to randomly flipping the signs of each paired sample.

```
> y.vec <- rep(1:19, 2) * rep(c(-1, 1), each = 19)
> y.vec

 [1]  -1  -2  -3  -4  -5  -6  -7  -8  -9 -10 -11 -12 -13 -14 -15 -16 -17 -18 -19
[20]   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19

> results2 <- safe(golub, y.vec, C.mat, local = "t.paired", Pi.mat = 1)
```

In Barry et. al. (2005), SAFE was applied to a Cox proportional hazards model for associating tumor gene expression to the survival of lung cancer patients. To include this functionality in `safe` The argument `local = "z.COXPH"` will conduct a univariate Wald test for each gene, with event times given as y.vec. This requires providing censoring indicators as a logical or numeric vector, `censor`, in the argument `args.local`:

```
> y.vec <- rexp(38)
> cens <- rep(0:1, c(30, 8))
> results2 <- safe(golub, y.vec, C.mat, local = "z.COXPH", Pi.mat = 1,
+     args.local = list(censor = cens))
```

In addition to these predefined local statistics, `safe` has been structured such that the user can specify other statistics. In creating a function for computing local statistics, it must take as input the matrix of expression data and response information as illustrated below; additional information can be passed as objects in the optional list, `args.local`. Here, we create a function for a one-sided Wilcoxon statistic for gene-specific increases in expression in the AML subtype (this choice of local statistic should not be confused with the default global statistic)

```
> local.Wilcoxon <- function(X.mat, y.vec, ...) {
+     return(function(data, trt = (y.vec == 1)) {
+         return(as.numeric(trt %*% apply(data, 1, rank)))
+     })
+ }
> results2 <- safe(golub, golub.cl, C.mat, Pi.mat = 1, local = "Wilcoxon")

> cbind(Student1 = round(results@local.stat[1:3],3),
+       Rank.Sum=results2@local.stat[1:3])

                            Student1 Rank.Sum
AFFX-HUMISGF3A/M97935_MA_at    2.502      269
AFFX-HUMISGF3A/M97935_MB_at    1.156      232
AFFX-HUMISGF3A/M97935_3_at    -0.110      194
```

As a resampling-based method, `safe` is computationally intensive, so considerations of efficiency should made in programming user-defined functions for local and global statistics. The above example, while simple, is much slower than the default run of `safe` because of the `apply` function. Likewise, for Barry et. al. (2005), a separate and faster function was written in C for solving the iterative solution to the univariate Cox proportional hazards model. Interfacing with C or another foreign language is highly suggested for intensive computational settings. A complete discussion of how to design and include user-defined functions will not be included in this vignette.

5 Alternative Global Statistics

In the above SAFE analyses, a functional category was compared to its complement set of genes through a Wilcoxon rank sum statistic. The merits of using rank-based statistics for functional analysis are discussed in more detail in Barry *et al.* (2005). However, the the

SAFE framework naturally extends to other statistics used in gene category analyses. This way one more properly account for gene correlation in testing categories (see Barry *et al.* 2008).

By default, `safe` conducts two-sided tests, whereby one takes the absolute value of local statistics such as a Student's t, before ranking genes. In this way, one can identify categories showing both consistent up-regulation, down-regulation, and also bi-directional differential expression. To conduct one-sided tests, one must specify `args.global = list(one.sided=T)` to consider only genes in the positive direction to be significant.

One popular way of examining categories is through "gene-list enrichment" methods, that were developed as *post hoc* means of testing once the genes with significant differential expression had been identified. These methods use a global statistics that only consider the dichotomous outcomes of gene-specific hypothesis tests, and typically use Fisher's Exact test, or Pearson's test for a difference in proportions. p-values are often times extremely anti-conservative under the false assumption of gene independence, which can lead to spurious results. For this reason, we have extended SAFE to these global statistics such that valid p-values can be obtained. In using the gene-list type global statistics, one must specify either the list length, as in the example below, or a (one- or two-sided) cut-off value:

```
> set.seed(12345)
> results2 <- safe(golub, golub.cl, C.mat, global = "Fisher", args.global = list(one.sided =
+     genelist.length = 200))

> results2

SAFE results:
  Local: t.Student
  Global: Fisher
  Method: permutation

           Size Num.Reject Emp.pvalue
KEGG:04640   61         10      0.017
KEGG:04970   37          7       0.02
KEGG:00561   12          3      0.026
KEGG:05120   34          6      0.027
KEGG:00590   17          4      0.031
KEGG:05144   32          6      0.034
KEGG:00860   15          4      0.039
KEGG:04110   57          9      0.044
KEGG:04142   53          8      0.049
```

The following calculation demonstrates the inappropriate p-value one would get from a naive application of Fisher's Exact test to KEGG:04640.

```
> 1 - phyper(12 - 1, 70, 3051 - 12, 200)

[1] 0.001391427
```

Alternatively, a one-sided cutoff value for local statistics is declared by the argument `args.global = list(one.sided = TRUE, genelist.cutoff=2.0)`. Further, the Pearson

test for difference in proportions (which is equivalent to a Chi-squared test) can be specified by the argument `global="Pearson"`, and the cutoff for the gene-list is specified in the same manners as shown above.

One can also substitute the average difference in local statistics as a measure of category-wide increases in differential expression using the argument `global="AveDiff"`. An alternative non-parametric 2-sample comparison that is also valid (albeit more computationally intensive) is the Kolmogorov-Smirnoff test; a computationally inefficient test can be specified as `global="Kolmogorov"`.

See `http://www.duke.edu/~dinbarry/SAFE/` for further examples that use these alternative global statistics.

6 Adjusting for multiple-testing in SAFE

As in standard gene-by-gene analyses, it is important to account for multiple comparisons when considering a set of categories. Since SAFE is a resampling-based test, permutation-based error rate methods have been incorporated into `safe` when applicable. Shown below are the results from the first example once multiple testing is accounted for using the Yekutieli-Benjamini method of estimating the *false discovery rate* (FDR).

```
> set.seed(12345)
> results <- safe(golub, golub.cl, C.mat, error = "FDR.YB", alpha = 0.25)

> results

SAFE results:
  Local: t.Student
  Global: Wilcoxon
  Method: permutation
  Error: FDR.YB

            Size Mean.Rank Emp.pvalue Adj.pvalue
KEGG:04110   57    1955.9      0.003     0.1753
KEGG:00860   15    2412.2      0.004     0.1753
```

NOTE: By default, when correcting for multiple testing, the cutoff for display changes from categories with empirical p-value less than 0.05 to those with adjusted p-values less than 0.1. The cutoff for display can be changed with the `alpha` argument in `safe`

As shown above, the most significant KEGG pathways are only marginally significant in their association to leukemia subtype after accounting for multiple comparisons through the FDR. In addition to the Yekutieli-Benjamini FDR estimate, `safe` can use the permutation algorithm to estimate the *family-wise error rate* with the Westfall-Young method (`error = "FWER.WY"`). Although we feel these two permutation-based procedures for controlling error are superior by empirically accounting for correlation among tests, one can also apply tradition methods including a Bonferroni correction or Holm's step-down procedure for the family-wise error rate (`error = "FWER.Bonf"`) or (`error ="FWER.Holm"`), and the Benjamini-Hochberg step-up procedure to control the false discovery rate (`error = "FDR.BH"`). These may be useful when comparing results from SAFE with other non-resampling-based procedures, or when using the bootstrap algorithms discussed in the following section.

7 Bootstrap-based tests in SAFE

In Barry *et al.* (2008), a bootstrap-based version of SAFE is proposed that is shown to generally be more powerful, and applicable to a wider set of experimental designs. Two basic methods of hypothesis testing are available: 1) The argument (`method="bootstrap"`) or (`method="bootstrap.t"`), will invoke pivot tests to look for the exclusion of a null value from Gaussian confidence intervals based on resampled estimates of the mean and variance of the global statistic; 2) alternatively (`method="bootstrap.q"`), will invoke tests based on the exclusion of a null value from the alpha-quantile interval of the resampled global statistic.

The following illustrates that more categories are identified as marginally significant under bootstrap-resampling version of SAFE.

```
> set.seed(12345)
> results2 <- safe(golub, golub.cl, C.mat, method = "bootstrap",
+                  error = "FDR.BH")

> results2

SAFE results:
  Local: t.Student
  Global: Wilcoxon
  Method: bootstrap
  Error: FDR.BH

           Size Mean.Rank Emp.pvalue Adj.pvalue
KEGG:00561   12    2086.6      2e-04     0.0099
KEGG:04920   27    1895.1      2e-04     0.0099
KEGG:04110   57    1955.9      2e-04     0.0099
KEGG:00970   16    2133.1      3e-04     0.0099
KEGG:05144   32    1953.0      4e-04     0.0125
KEGG:04966   10    2312.9      9e-04     0.0223
KEGG:00860   15    2412.2     0.0013     0.0261
KEGG:05340   25    1825.5     0.0018     0.0317
KEGG:04621   22    1896.4     0.0021     0.0326
KEGG:05146   49    1821.9     0.0027     0.0388
KEGG:04640   61    1768.6     0.0045     0.0534
KEGG:03050   23    2108.7     0.0045     0.0534
```

Based on the requirements for bootstrap-based hypothesis testing (see Barry *et al.* 2008 for explanation), they can only be performed using (`global %in% c("Wilcoxon", "AveDiff", "Pearson")`). Further the permutation-based error rate estimates are no longer applicable, so that the options available are (`error %in% c("FDR.BH", "FWER.Bonf", "FWER.Holm", "none")`).

By default, the data are resampled 1000 times when selecting `method %in% c("permutation", "bootstrap.q")` though often times $>$ 10-fold more resamples are suggested if there are several hundred categories being investigated. The Gaussian bootstrap-based test has the added advantage that empirical p-values are not bounded by the total number of resamples taken. Thus, small p-values can be obtained without intensive computational effort;

this is of particular importance when overcoming stringent correction for high degrees of multiple-testing. As such, 200 resamples are taken for (`method = "bootstrap"`), and have been demonstrated to provide sufficient error control.

Also, permutation-based p-values for local statistics are no longer obtained under bootstrap resampling. Instead, empirical p-values can be obtained using the exclusion of 0 from quantile intervals with (`args.local = list(boot.test = "q")`) and Gaussian intervals with (`args.local = list(boot.test = "t")`). A null value of 0 relates to no differential expression in the supplied local statistics, however this must be reasonable for any user-defined statistics (*e.g.* it does not apply for a Kolmogorov-Smirnov test statistic).

8 Sources of Functional Categories

In the above sections, functional categories were derived from KEGG pathways as provided in the package `hu6800`. Functional categories can also be derived from other sources of information in the same package. The Protein Families database can also be used to generate categories of genes that share homologous domains:

```
> results2 <- safe(golub, golub.cl, platform = "hu6800", annotate = "PFAM",
+     min.size = 10, method = "bootstrap")

Building PFAM categories from hu6800PFAM
  Categories completed:20% 40% 60% 80% 100%
  68 catgories formed
100 bootstrap resamples completed
200 bootstrap resamples completed

> results2

SAFE results:
  Local: t.Student
  Global: Wilcoxon
  Method: bootstrap

        Size Mean.Rank Emp.pvalue
PF00227   12    2201.0     0.0015
PF00307   20    2121.8     0.0037
PF00089   18    1859.6     0.0038
PF00628   11    1956.1     0.0104
PF00048   17    1907.5     0.0122
PF02518   14    1868.2     0.0262
PF00433   17    1815.1     0.0263
```

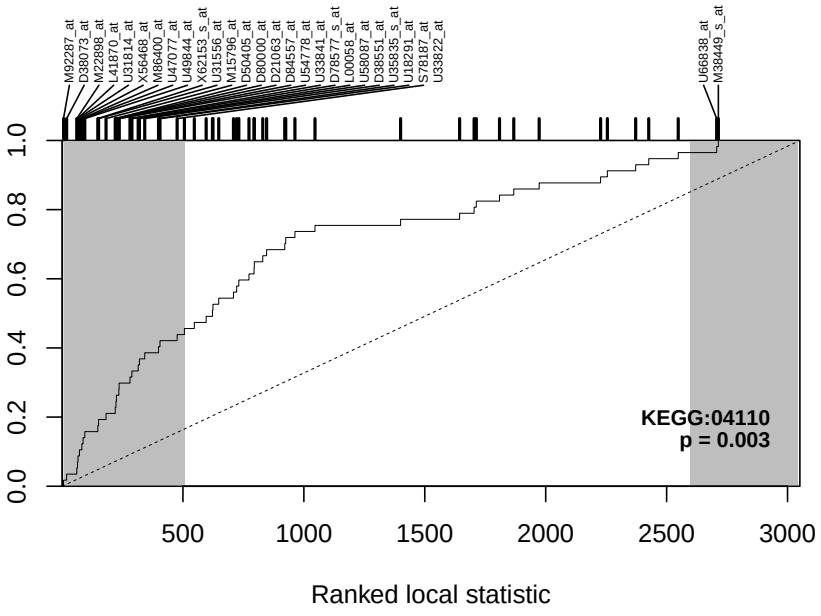
Gene Ontology is also available from `hu6800` and other Bioconductor metadata packages. `annotate = "GO.ALL"` will form categories from all three ontologies, while "GO.CC", "GO.BP", and "GO.MF" will work with Cellular Compartment, Biological Process and Molecular Function respectively. It is important to note that in the hierarchical structure of the GO vocabularies, a gene category is generally thought of as containing the set of genes directly annotated to a term, and also to any terms beneath it in the ontology. The C.matrix of each can be externally built with the `getCmatrix` function as follows (in a much more efficient manner than in SAFE 1.0).

```
> GO.list <- as.list(hu6800GO2ALLPROBES)
> C.mat.CC <- getCmatrix(keyword.list = GO.list, GO.ont = "CC",
+     present.genes = dimnames(golub)[[1]], min.size = 10, max.size = 200)

  Categories completed:20% 40% 60% 80% 100%
  229 catgories formed

> results2 <- safe(golub, golub.cl, C.mat.CC, method = "bootstrap")

100 bootstrap resamples completed
200 bootstrap resamples completed
```

9 Plotting SAFE results

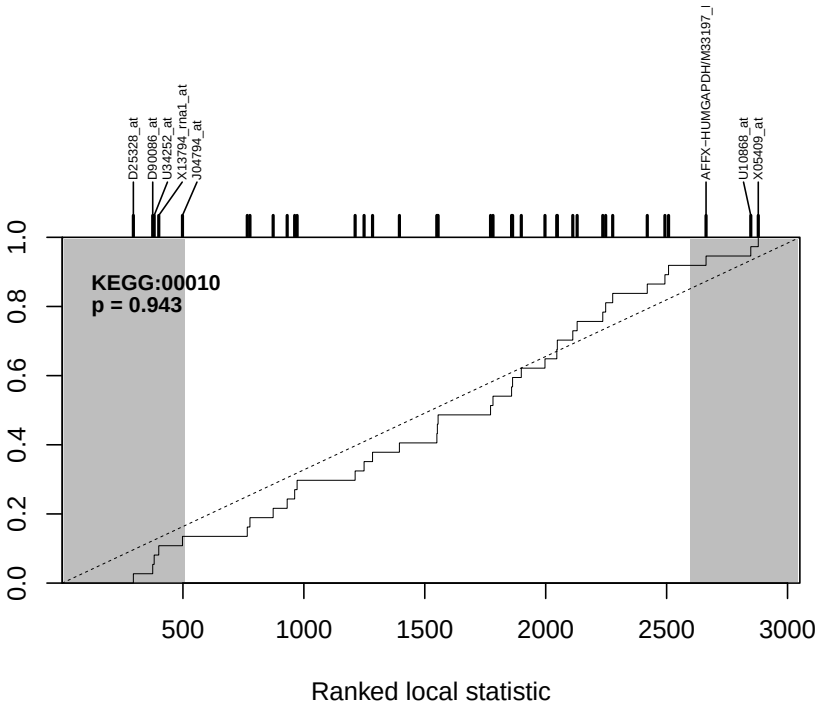
For a single category, we have proposed that the differential expression of genes be plotted as a SAFE-plot (Barry *et al.*, 2005). Shown below is the SAFE-plot for the most significant KEGG pathway, which is the default output of `safeplot` when a object of class *SAFE* is provided.

```
> safeplot(results)
```



SAFE-plots of other categories can be generated with the argument `cat.name`, as shown below for a non-significant category.

```
> safeplot(results, cat.name = "KEGG:00010")
```

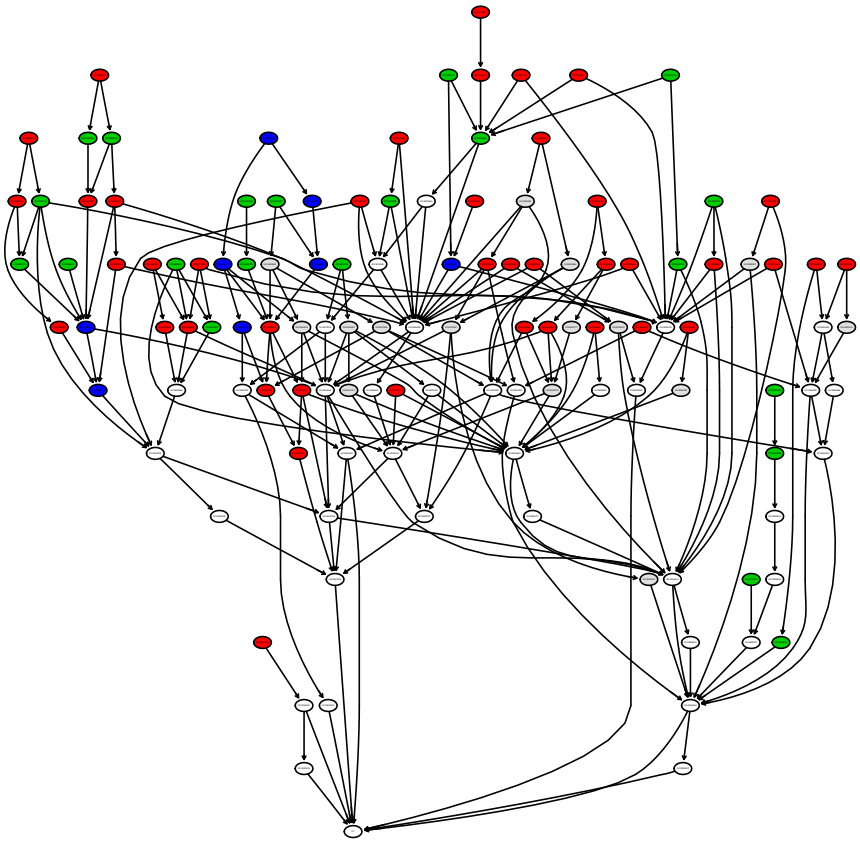


SAFE-plots show the cumulative distribution function (CDF) for the ranked local statistics from a given category (solid line). A significant category will have more extreme associations to the response of interest than its complement, resulting in either a right-ward, left-ward, or bidirectional shift in the CDF away from the unit line (dashed line). The shaded regions of the plot correspond to the genes that pass a nominal level of significance (empirical p-values ≤ 0.05 by default). Also, the genes in the category are shown as tick marks along the top of the graph, and depending on the category size, either all genes in the category are labeled, or only ones in the shaded region of the graph. Thus SAFE-plots show that the KEGG pathways 00860 and 00590 show upregulation in AML on average, while 00970 shows downregulation in AML on average, and 00010 shows no consistent trend of differential expression.

Finally, Gene Ontology is a unique structured vocabulary where genes are annotated from broad to narrow levels of classification in a directed acyclic graph (DAG). As such, many categories are highly related in their gene membership, and visualizing results across the ontology can be useful in ascertaining the relationship among multiply significant categories. The following function interacts with the `GOstats` and `Rgraphviz` packages in order to overlay SAFE results onto the DAG structure in a color-metric manner. By default, nodes with unadjusted p-values less than 0.001 are drawn in blue; less than 0.01 are drawn in green; and less than 0.1 are drawn in red. User-defined cutoffs for the three colors can be specified using the argument `color.cutoffs`. Further illustrations are provided in example

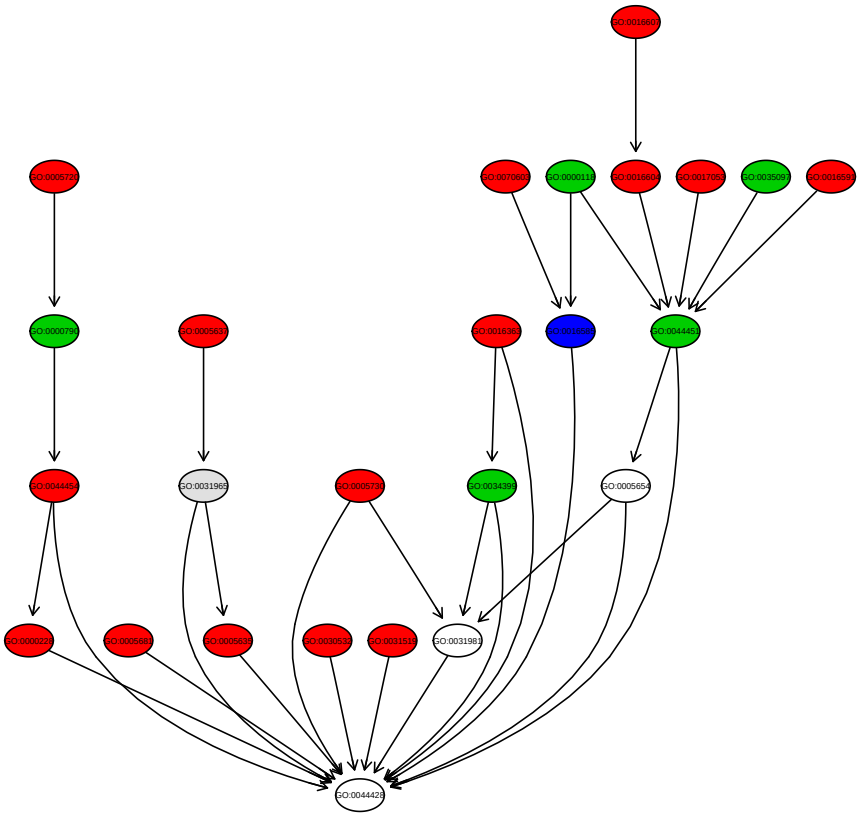
scripts available at `http://www.duke.edu/~dinbarry/SAFE/`

```
> safedag(results2, filter = 1)
```



And one can also zoom in on parts of the DAG by specifying a node to be the top of the graph.

```
> safedag(results2, filter = 1, top = "GO:0044428")
```



10 References

- Barry, W.T., Nobel, A.B. and Wright, F.A. (2005) 'Significance Analysis of functional categories in gene expression studies: a structured permutation approach', *Bioinformatics*, **21**(9), 1943–1949.

- Barry, W.T., Nobel, A.B. and Wright, F.A. (In press) 'A Statistical Framework for Testing Functional Categories in Microarray Data', *Annals of Applied Statistics*.

- Virtaneva, K.I., Wright, F.A., Tanner, S.M., Yuan, B., Lemon, W.J., Caligiuri, M.A., Bloomfield, C.D., de laChapelle, A., & Krahe, R. (2001) 'Expression profiling reveals fundamental biological differences in acute myeloid leukemia with isolated trisomy 8 and normal cytogenetics'*Proc Natl Acad Sci U S A* **98**(3), 1124–1129.

- Golub, T.R., Slonim, D.K., Tamayo, P., Huard, C., Gaasenbeek, M., Mesirov, J.P., Coller, H., Loh, M.L., Downing, J.R., Caliguiri, 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**, 531–537.

- Yekutieli, D. and Benjamini, Y. (1999) 'Resampling-based false discovery rate controlling multiple test procedures for correlated test statistics' *J Statist Plann Inference* **82**, 171–196

• Westfall, P.H. and Young, S.S. (1989) 'P-value adjustment for multiple tests in multivariate binomial models', *J Amer Statist Assoc* **84**, 780–786