

# Significance Analysis of Function and Expression

William T. Barry  
wbarry@bios.unc.edu

April 25, 2007

## 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 permutation-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 a more recent publication from Barry *et al.* (2005). It is suggested that all users read Barry *et al.* (2005) to understand the SAFE terminology and principles in greater detail. We also ask that Barry *et al.* (2005) be cited in publications using **safe**.

Here, we will focus our attention on the implementation of **safe** using datasets and annotations available from the Bioconductor packages listed below.

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

## 2 Necessary components of SAFE

Every SAFE analysis requires three elements from a dataset: gene expression data, a response vector associated with the samples, and the category assignments for genes on the array.

The expression data should be in the form of an  $m \times n$  matrix, where appropriate normalization and other preprocessing 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. Gene names (probeset IDs from the hu6800 affymetrix array) are available from the third column of the matrix **golub.gnames**. The comparison of interest will be between AML and ALL tumors subtypes. Tumor classification of samples is contained in the vector **golub.c1** ( AML = 1, ALL = 0 ). Section

```
> golub.cl
```

For this example, 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 10 genes among the 3,051 in the `golub` dataset, resulting in 65 categories being tested.

It is strongly suggested that appropriate names are given to these elements so the output from `safe` is properly labeled.

The following demonstrates a SAFE analysis of KEGG pathways in the AML/ALL dataset, which uses the default arguments of `safe`. A seed is specified so that the default `safe` output is reproducible for this illustration, but is not recommended for general use.

```
SAFE results:
  Local: t.Student
  Global: Wilcoxon
```

The basic output from **safe** is displayed above. The SAFE framework for testing gene categories is a 2-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 shown above, 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 across a KEGG pathway is determined with a Wilcoxon rank sum as the default global statistic.

The output of `safe` is an object of class `SAFE` that contains both local and global results along with the information necessary for plotting. Objects of class `SAFE` will automatically print the results for categories that attain a given level of significance (described in more detail below.) Here, the categories with empirical p-values  $\leq 0.05$  are printed. For each category, the number of annotated genes in the dataset is displayed along with the global statistic and its empirical p-value.

An integral part of a SAFE analysis is the permutation-based testing. Empirical p-values are calculated for each category through reassignment of the response vector. In this manner, the unknown correlation among genes is conserved across permutation and thereby accounted for in tests. In order to specify permutations in `safe`, the argument `Pi.mat` can have either an integer value or matrix passed (Note: if left unspecified, `safe` will automatically generate 1000 random permutations). If an integer is passed, a corresponding number of random permutations of the response vector are generated. Else, a matrix of permissible permutations can be created through the function `getPImatrix`. Using `getPImatrix`, one can hold blocking variables constant across permutation. In later versions of `safe`, exhaustive permutations of moderately sized datasets will be possible, producing exact empirical p-values. Also, by storing the matrix generated by `getPImatrix`, one can reproduce results without having to specify seeds.

```
> PI <- getPImatrix(y.vec = golub.cl, K = 100)
> PI[1:2, ]
```

	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]	[,7]	[,8]	[,9]	[,10]	[,11]	[,12]	[,13]	[,14]
[1,]	1	2	3	4	5	6	7	8	9	10	11	12	13	14
[2,]	15	22	25	35	9	21	8	10	29	26	28	4	19	37
	[,15]	[,16]	[,17]	[,18]	[,19]	[,20]	[,21]	[,22]	[,23]	[,24]	[,25]	[,26]		
[1,]	15	16	17	18	19	20	21	22	23	24	25	26		
[2,]	6	18	34	2	16	7	30	14	13	12	33	17		
	[,27]	[,28]	[,29]	[,30]	[,31]	[,32]	[,33]	[,34]	[,35]	[,36]	[,37]	[,38]		
[1,]	27	28	29	30	31	32	33	34	35	36	37	38		
[2,]	38	31	20	11	23	36	27	3	32	1	5	24		

The first row of `PI` corresponds to the observed data, and will thus contain the ordered integers from 1 to  $n$ . All subsequent rows give the reordered column numbers for that permutation.

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 permutation-based test, resampling-based error rate methods have been incorporated into `safe`. Shown below are the results for the KEGG pathways once multiple testing is accounted for using the Yekutieli-Benjamini method of estimating the *false discovery rate* (FDR).

```

> results2 <- safe(golub, golub.cl, C.matrix, error = "FDR.YB",
+   alpha = 0.2)

> results2

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

[1] "No categories were significant at FDR.YB < 0.2"

> min(results2@global.error)

[1] 0.2389378

```

As shown above, no KEGG pathways appear to be significantly associated with leukemia subtype after accounting for multiple comparisons. The minimum error rate, which can be interpreted as an adjusted p-value for the most extreme category, is also printed above. The argument `alpha` allows the user to set the maximum error rate (or nominal p-value if `error = "none"`) for showing the `safe` results. In addition to the Yekutieli-Benjamini FDR estimate, `safe` can estimate the *family-wise error rate* using the Westfall-Young method with the argument, `error = "FWER.WY"`.

## 4 Experimental Designs and Local Statistics

The example above focuses on a simple two-sample comparison, as one of several experimental designs that `safe` can automatically accommodate. In addition to the internal local statistics for generic comparisons, one can also employ user-defined functions in `safe` to extend it further. The following examples illustrate the arguments needed for alternative designs and statistics, and also show the slots in a *SAFE* object for the observed local statistics and their respective empirical p-values. Results for the first four genes are compared to the default run of `safe` in Section 3:

```

> results@local

[1] "t.Student"

> cbind(Stat = results@local.stat, Emp.p = results@local.pval)[1:4,
+   ]

```

	Stat	Emp.p
AFFX-HUMISGF3A/M97935_MA_at	2.5021067	0.017
AFFX-HUMISGF3A/M97935_MB_at	1.1561671	0.265
AFFX-HUMISGF3A/M97935_3_at	-0.1099865	0.911
AFFX-HUMRGE/M10098_5_at	-0.2726576	0.786

In two-sample comparisons, the response vector can either be given as a (0,1) vector, or a character vector with two unique elements.

```
> y.vec <- rep(c("ALL", "AML"), table(golub.cl))
> y.vec

[1] "ALL" "ALL" "ALL" "ALL" "ALL" "ALL" "ALL" "ALL" "ALL" "ALL" "ALL" "ALL"
[13] "ALL" "ALL" "ALL" "ALL" "ALL" "ALL" "ALL" "ALL" "ALL" "ALL" "ALL" "ALL"
[25] "ALL" "ALL" "ALL" "AML" "AML" "AML" "AML" "AML" "AML" "AML" "AML" "AML"
[37] "AML" "AML"
```

```
> safe(golub, y.vec, C.matrix, Pi.mat = 1)@local.stat[1:4]
```

```
[1] "Warning: y.vec is not (0,1), thus 1 == ALL"
AFFX-HUMISGF3A/M97935_MA_at AFFX-HUMISGF3A/M97935_MB_at
-2.5021067 -1.1561671
AFFX-HUMISGF3A/M97935_3_at AFFX-HUMRGE/M10098_5_at
0.1099865 0.2726576
```

It is important to note that when a character vector is passed to **safe** as the response, the assignment of the first array becomes the first group in the two sample comparison, and is printed as a warning. For this reason, the sign of the t-statistics has flipped in the above output. Since we are only concerned with the observed statistics in the following illustrations, the permutation testing is bypassed by using the argument ( $\text{Pi.mat} = 1$ ).

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.

```
> safe(golub, golub.cl, C.matrix, local = "t.Welch", Pi.mat = 1)@local.stat[1:4]
```

```
AFFX-HUMISGF3A/M97935_MA_at AFFX-HUMISGF3A/M97935_MB_at
1.75919522 0.90985764
AFFX-HUMISGF3A/M97935_3_at AFFX-HUMRGE/M10098_5_at
-0.09802592 -0.33896286
```

**safe** is also able to compute the SAM statistic from Tusher *et al.* (2001) for 2-sample comparisons (with unequal variances). It can be noted that the fudge factor  $s_0$  in the modified t-statistic is automatically printed by **safe**.

```
> safe(golub, golub.cl, C.matrix, local = "t.SAM", Pi.mat = 1)@local.stat[1:4]
```

```
[1] "Alpha = 0 ; s_0 = 0.0606485918851795"
AFFX-HUMISGF3A/M97935_MA_at AFFX-HUMISGF3A/M97935_MB_at
1.4458277 0.7259838
AFFX-HUMISGF3A/M97935_3_at AFFX-HUMRGE/M10098_5_at
-0.0755108 -0.3022926
```

For multi-class designs, response vectors should be character or numeric vectors with unique values for each group. If a response vector with more than two unique elements is given, an ANOVA F-statistic is computed by default; or an ANOVA test can be specified with the argument ( $\text{local} = \text{"f.ANOVA"}$ ). Note that an ANOVA F-statistic for 2 classes is equivalent to a squared Student's t-statistic.

```
> safe(golub, golub.cl, C.matrix, local = "f.ANOVA", Pi.mat = 1)@local.stat[1:4]
```

AFFX-HUMISGF3A/M97935_MA_at	AFFX-HUMISGF3A/M97935_MB_at
6.26053776	1.33672239
AFFX-HUMISGF3A/M97935_3_at	AFFX-HUMRGE/M10098_5_at
0.01209703	0.07434216

Lastly, simple linear regression can be done with the argument (`local = "t.LM"`). Again, the t-statistic for the slope parameter is equivalent to a Student's t-test when `y.vec` is a vector of 0's and 1's.

```
> safe(golub, golub.cl, C.matrix, local = "t.LM", Pi.mat = 1)@local.stat[1:4]
```

AFFX-HUMISGF3A/M97935_MA_at	AFFX-HUMISGF3A/M97935_MB_at
2.5021067	1.1561671
AFFX-HUMISGF3A/M97935_3_at	AFFX-HUMRGE/M10098_5_at
-0.1099865	-0.2726576

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 covariate information as specified in section 2. The following is a quick illustration using a one-sided Wilcoxon statistic for increased expression in the AML subtypes as the local statistic (this choice of local statistic should not be confused with the default global statistic, which also happens to be a Wilcoxon rank sum)

```
> local.Wilcoxon <- function(X.mat, y.vec) {
+   return(function(data, trt = (y.vec == 1)) {
+     return(as.vector(trt %*% apply(data, 1, rank)))
+   })
+ }
```

```
> safe(golub, golub.cl, C.matrix, Pi.mat = 1, local = "Wilcoxon")@local.stat[1:4]
```

AFFX-HUMISGF3A/M97935_MA_at	AFFX-HUMISGF3A/M97935_MB_at
269	232
AFFX-HUMISGF3A/M97935_3_at	AFFX-HUMRGE/M10098_5_at
194	198

It should be noted that as a permutation-based method, **safe** is a computationally intensive function. At the end of the following section, the computation times of different SAFE analyses will be tabulated. Consideration should be made in creating 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, in Barry et. al. (2005), **safe** was extended to the Cox proportional hazards model which has an iterative solution. Interfacing with C or another foreign language is highly suggested for such extreme computational settings. A complete discussion of how to include user-defined functions, and the optimal way of designing them 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 non-parametric Wilcoxon rank sum test. The

merits of using non-parametric tests for functional categories are discussed in more detail in Barry *et al.* (2005). An alternative non-parametric 2-sample comparison that would also be valid (albeit more computationally intensive) is the Kolmogorov-Smirnoff test, and can be specified in **safe** as follows.

```
> results2 <- safe(golub, golub.cl, C.matrix, Pi.mat = 1, global = "Kolmogorov")
> results2["KEGG00590"]
```

SAFE results:

```
Local: t.Student
Global: Kolmogorov
```

	Size	Stat
KEGG00590	19	53.396

To limit the computation in this vignette, permutations are not done in the above example. It is strongly suggested that users interested in Kolmogorov-Smirnoff type tests run the above statement with 1000 permutations and compare the results to the default settings.

Although we favor non-parametric global statistics in **safe**, the function can also be used to obtain permutation-based p-values for the popular Hypergeometric tests of a functional categories' representation in a genelist. For instance, suppose we consider the list of genes that have Student's t-statistics more extreme than -3 and 3. The following illustration also demonstrates how necessary information is passed to local and global statistics. Lists of additional information can be provided through the arguments (**args.global** and **args.local**). In this manner, the global statistic is specified to be two-sided (*i.e.* take the absolute value of local statistics), and the criterion for inclusion in the genelist is given.

```
> args <- list(one.sided = F, cut.off = 3)
> results2 <- safe(golub, golub.cl, C.matrix, Pi.mat = 1, global = "genelist",
+   args.global = args)
> sum(abs(results@local.stat) > 3)

[1] 552
> results2["KEGG00590"]
```

SAFE results:

```
Local: t.Student
Global: genelist
```

	Size	Stat
KEGG00590	19	7

In the above output, the statistic is the number of genes within the category that pass the cut-off for inclusion in the genelist. As discussed in Barry *et al.* (2005), the empirical p-value is more appropriate than what one would obtain from the assumed hypergeometric distribution because of the unknown

correlation among the genes that is uniquely accounted for in **safe**. Users interested in applying this global statistic should run the above statement with permutations to compare results.

The following table provides the computation times of **safe** using the Golub dataset and KEGG pathways with the arguments given in the sections above. To quantify the computational times, the **safe** examples were run interactively with measurements from **Sys.time** on two different machines: a 3GHz Pentium processor running Windows XP and R\_2.0.1, and a 3GHz Xeon processor running Red Hat Enterprise 3 and R\_2.0.0. Time is given in minutes in the following table.

Changes from the default settings	PC machine	UNIX machine
Default <b>safe</b> settings	0.37	0.65
error = "FDR.YB"	0.38	0.68
local = "Wilcoxon"	8.95	10.13
global = "Kolmogorov"	5.93	6.65
global = "genelist"	0.40	0.68

## 6 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 and databases. One type of category of increasing interest is Gene Ontology, whose annotation is also available from **hu6800**. 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. Although **hu6800** provides this information in a slightly different format than KEGG, as a list of GO terms instead of a list of probesets, the function **getCmatrix** can be used to get the transpose of the matrix of interest. The following steps demonstrate one way to get a GO-based C matrix for the **golub** dataset.

To reduce computation, we will restrict ourselves to the first 20 GO terms,

```
> GO.list <- as.list(hu6800GO2ALLPROBES)[1:20]
```

We then "trick" **getCmatrix** into generating the transpose of the category assignments.

```
> Transpose <- getCmatrix(GO.list)
> dim(Transpose)
```

```
[1] 20 210
```

Next, we remove the genes on the **hu6800** array that did not appear in **golub**, and insert the remaining transposed assignments into an appropriately sized C matrix.

```
> Transpose <- Transpose[, dimnames(Transpose)[[2]] %in% dimnames(golub)[[1]]]
> dim(Transpose)
```

```
[1] 20 84
```



```
> C.matrix <- matrix(0, dim(golub)[[1]], dim(Transpose)[[1]])
> dimnames(C.matrix) <- list(dimnames(golub)[[1]], dimnames(Transpose)[[1]])
> C.matrix[match(dimnames(Transpose)[[2]], dimnames(golub)[[1]]),
+       ] <- t(Transpose)
```

Lastly, we can filter out categories that would be too small to be of interest; here we set a minimum category size of 10 genes.

```
> C.matrix <- C.matrix[, apply(C.matrix, 2, sum) > 9]
> dim(C.matrix)

[1] 3051    3
```

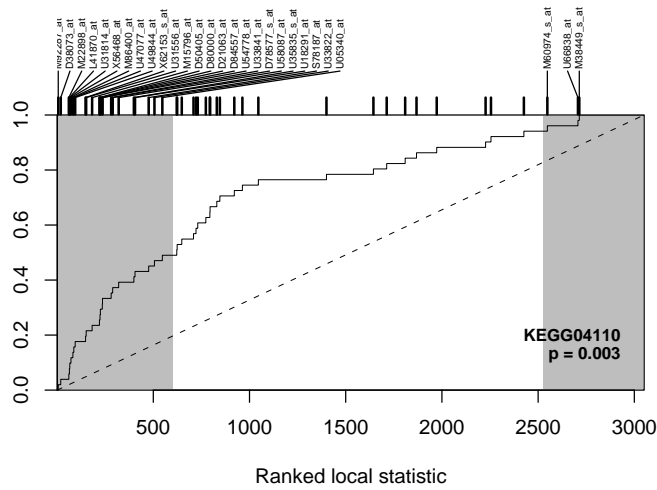
The output above demonstrates that only 4 of the first 20 GO terms were large enough for consideration. In their entirety, 753 GO terms are linked to at least 10 genes in `golub`. Other steps would be necessary to restrict an analysis to a single ontology (e.g. Biological Processes), or some other criterion.

For Affymetrix arrays, the annotation of probesets is also available in NetAffx files ([www.affymetrix.com](http://www.affymetrix.com)), and provides GO terms, Pfam domains, and GenMAPP pathways among other things. Basic functions in R can extract the information from the NetAffx files to be used in `getCmatrix`.

## 7 Plotting SAFE results

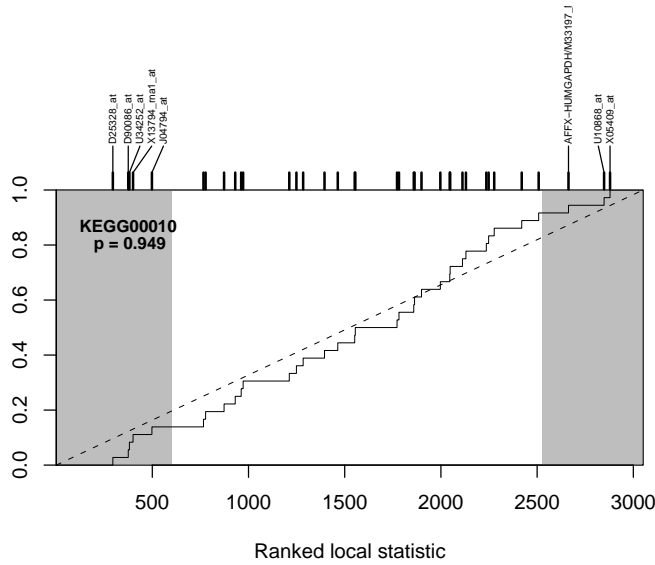
For a single category, we have proposed that the differential expression of genes be plotted as a SAFE-plot (Barry *et al.*, 2005). Shown below is the SAFE-plot for the most significant KEGG pathway, which is the default output of `safeplot` when a object of class *SAFE* is provided.

```
> safeplot(results)
```



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

```
> safepplot(results, cat.name = "KEGG00010")
```



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  $\leq 0.05$  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. In our 2-sample comparison of AML and ALL, genes on the right side of the SAFE-plot are upregulated in AML tumors relative to ALL, and genes on the left side are downregulated. 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.

## 8 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*, In press.
- 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., 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**, 531–537.
- Tusher, V.G., Tibshirani, R., and Chu, G. (2001) ‘Significance Analysis of Microarrays Applied to the Ionizing Radiation Response’, *Proc Natl Acad Sci U S A* **98**, 5116–5121.
- 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