Significance Analysis of Function and Expression

William T. Barry * Alexander B. Sibley Fred Wright Yihui Zhou

October 13, 2015

1 Introduction

This vignette demonstrates the utility and flexibility of the R package `safe` in conducting tests of functional categories for gene expression studies. Significance Analysis of Function and Expression (SAFE) is a resampling-based method that is applicable to many different experimental designs and functional categories. SAFE extends and builds on an approach first employed in Virtaneva *et al.* (2001), and defined more rigorously in Barry *et al.* (2005 and 2008). Gatti *et al.* (2010) showed that many applications for pathway analysis continue to utilize methods which are grossly anti-conservative, and would therefore lead to a very high false-positive rate in the literature. Lastly, in Zhou *et al.* (2013), we developed a series of novel analytical approximations of permutation-based tests of pathways. These have improved properties over the use of random sampling in selecting permutations, and greatly reduce the computational requirements when inferences are based on the extreme tails of empirical distributions. It is suggested that users refer to these publications to understand the SAFE terminology and principles in greater detail.

2 Citing `safe`

When using the results from the `safe` package, please cite:

> 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.

and

> Barry, W.T., Nobel, A.B. and Wright, F.A. (2008) 'A Statistical Framework for Testing Functional Categories in Microarray Data', *Annals of Applied Statistics*, **2**(1), 286–315.

The above articles describe the methodological framework behind the `safe` package.

3 Updates since version 2

The following lists summarize the changes and extended capability of `safe` that are included in version 3. Examples of their implementation are illustrated in subsequent sections, and additional details are given in help documents.

*bbarry@jimmy.harvard.edu

3.1 Major extensions

- A new method for including covariates in `safe` is implemented with the option argument `Z.mat`, and relies on the internal function `getXYresiduals`. The extension is discussed in Zhou *et al.* (2013), and an example is given in subsection 6.7.
- A new approach for pathway-analyses with right-censored time-to-event data is also discussed in Zhou *et al.* (2013), and an example is given in subsection 6.6. The computationally intensive method included in version 2 (`local = "z.COXPH"`) is depreciated and has been removed.
- The internal function `getCmatrix` is updated for improved efficiency and a new optional argument, `by.gene = TRUE`, to consider pathways at the gene-level instead of the probeset-level.
- New functionality is added to `safe` to implement parallel processing. Usage instructions and examples of improved execution times are given in section 10.
- For basic experimental designs, `safe` automatically switches to exhaustive permutation when there are fewer than what is specified by the default or user. The default for `method = "permutation"` remains `Pi.mat = 1000`.
- `safe` is modified to include the novel analytic approximations to permutation-testing proposed in Zhou *et al.* (2013) using a dependent package, `safeExpress`. Users interested in this method should contact any of the co-authors for the package source, which can be installed on any operating system.

3.2 Minor changes

- Names are attached to all slots of object of class *SAFE*.
- `safe.toptable` is added for tabulating the output from `safe`.
- Several new options are included in `safeplot` for visualizing the output from `safe`.
- The user-controlled argument `epsilon = 1e-10` corrects a numerical precision issue when computing empirical p-values in small data sets ($n < 15$).
- The default manner for accounting for multiple testing is switched from `error = "none"` to `error = "FDR.BH"` to adjust p-values by the Benjamini-Hochberg (1995) estimate of the *false discovery rate*.

4 SAFE implementation and output

The following Bioconductor packages are required for applying `safe` to an Affymetrix breast cancer dataset from Miller *et al.* (2005).

```
> library(breastCancerUPP)
> library(hgu133a.db)
> library(safe)
```

Every SAFE analysis requires as input three elements from an experiment: (1) gene expression data, (2) phenotype/response information associated with the samples, and (3) category assignments that are either pre-built or generated from Bioconductor annotation packages for the array platform.

The expression data should be in the form of an $m \times n$ matrix (m = the number of features in the array platform, n = the number of samples), where appropriate normalization and other pre-processing steps have been taken. It should be noted that missing values are not allowed in the expression data, and must be imputed prior to analysis.

This tutorial will use the ExperimentData package `breastCancerUPP`, containing the object `upp`, an ExpressionSet of normalized expression estimates (for 251 samples) that are concatenated from the Affymetrix U133A and U133B platforms.

One sample characteristic of interest is p53 mutation status, that we append to the phenotype data for `upp`. p53 mutation status is taken from the NCBI's Gene Expression Omnibus (Edgar *et al.*, 2002), accession GSE3494 (Miller *et al.*, 2005), where p53+ = 1 and p53− = 0.

```
> data(upp)
> ex.upp <- exprs(upp)
> p.upp <- pData(upp)
> data(p53.stat)
> p.upp <- cbind(p.upp, p53 = p53.stat$p53)
```

For the purposes of this vignette, the phenotype and expression data are restricted to only those samples indicated as Grade 3, and the expression matrix is reduced to only the non-control probesets on the Affymetrix hgu133a array.

```
> grade.3 <- which(p.upp$grade == 3)
> p3.upp <- p.upp[grade.3,]
> genes <- unlist(as.list(hgu133aSYMBOL))
> drop <- grep("AFFX", names(genes))
> genes <- genes[-drop]
> e3.upp <- ex.upp[match(names(genes), rownames(ex.upp)),
+                  grade.3]
> table(p53 = p3.upp$p53)

p53
 0  1
23 31
```

Probeset IDs are necessary as row names in `e3.upp` for building gene categories. Here, the functional categories of interest are REACTOME pathways and are identified internally by the `safe` function; the process of generating categories is discussed in more detail in section 5.

```
> set.seed(12345)
> results <- safe(e3.upp, p3.upp$p53, platform = "hgu133a.db",
+                 annotate = "REACTOME", print.it = FALSE)

Building categories from reactome.db by ENTREZID1640 categories formed
```

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 feature-by-feature manner, and a "global" statistic measures the extent of association in features assigned to a category relative to the complement. The default local statistic for the two-sample comparison of p53+ and p53− is the Student's t-statistic, and the default global statistic is the Wilcoxon rank sum. Empirical p-values for local and global statistics are calculated by permutation.

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 categories that attain a certain level of significance. In addition, the function `safe.toptable` is included in version 3.0 of the `safe` package to return annotated results as a `data.frame` for categories with the strongest association to response/phenotype. This includes (a) category name; (b) the category size; (c) the global statistic; (d) nominal empirical p-values; (e) adjusted p-values; and (f) descriptions of Gene Ontology (GO), REACTOME genesets, or Protein Families (PFAM), if available.

```
> safe.toptable(results, number = 10)
```

	GenesetID	Size	Statistic	P.value	Adj.p.value	Description
1	REACTOME:5607763	26	412085	0.001	0.4100	CLEC7A (Dectin-1) induces NFAT activation
2	REACTOME:3065676	7	136374	0.001	0.4100	SUMO is conjugated to E1 (UBA2:SAE1)
3	REACTOME:977347	3	65424	0.001	0.4100	Serine biosynthesis
4	REACTOME:110314	40	609999	0.002	0.4100	Recognition of DNA damage by PCNA-containing replication complex
5	REACTOME:69183	20	327234	0.002	0.4100	Processive synthesis on the lagging strand
6	REACTOME:69166	18	291188	0.003	0.4100	Removal of the Flap Intermediate
7	REACTOME:5625900	21	321557	0.004	0.4100	RHO GTPases activate CIT
8	REACTOME:174414	14	227092	0.004	0.4100	Processive synthesis on the C-strand of the telomere
9	REACTOME:3065679	9	160028	0.004	0.4100	SUMO is proteolytically processed
10	REACTOME:1614603	3	61033	0.004	0.4100	Cysteine formation from homocysteine

NOTE: As in standard feature-by-feature analyses, it is of critical importance to account for multiple comparisons when considering a number of categories simultaneously. By default, `safe` provides Benjamini-Hochberg (1995) adjusted p-values to control the false-discovery rate. Several other options for correcting for multiple testing are discussed in detail in Section 8.

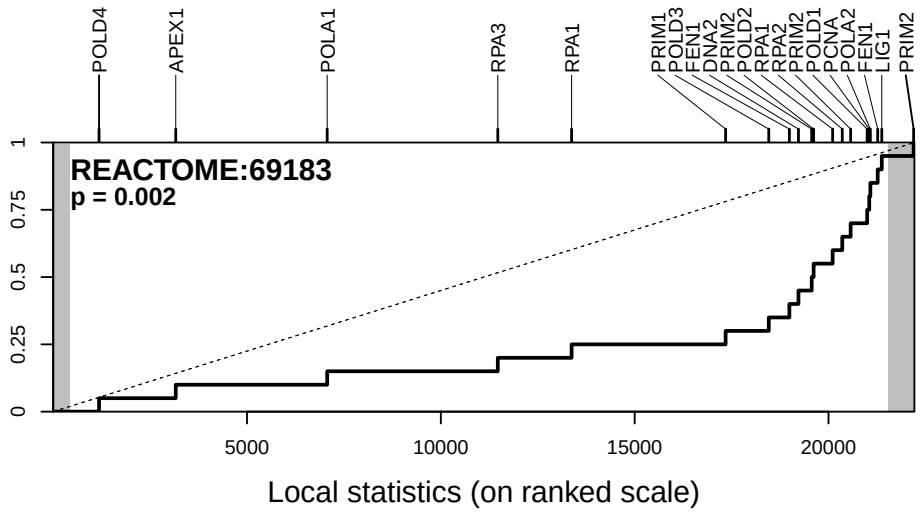
Feature-specific results within a category can be extracted by the `gene.results` function. This is useful for investigators interested in knowing which members of a category are contributing to its significance. The following example demonstrates how the direction and magnitude of differential expression 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 = "REACTOME:69183", gene.names = genes)

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

REACTOME:69183 consists of 20 gene features

Upregulated Genes
-----------------
            Gene.Names Local.Stat Emp.pvalue
```

```
205628_at        PRIM2      3.593      0.002
202726_at         LIG1      2.031      0.050
204768_s_at       FEN1      1.977      0.057
203422_at        POLD1      1.840      0.065
204441_s_at      POLA2      1.872      0.070
201202_at         PCNA      1.857      0.080
215709_at        PRIM2      1.666      0.092
201756_at         RPA2      1.602      0.107
201528_at         RPA1      1.526      0.124
215708_s_at      PRIM2      1.379      0.173
213647_at         DNA2      1.295      0.175
201115_at        POLD2      1.390      0.185
204767_s_at       FEN1      1.246      0.225
212836_at        POLD3      1.130      0.249
205053_at        PRIM1      0.921      0.358
201529_s_at       RPA1      0.328      0.740
209507_at         RPA3      0.082      0.933

Downregulated Genes
-------------------
            Gene.Names Local.Stat Emp.pvalue
202996_at        POLD4     -1.652      0.118
210027_s_at      APEX1     -1.106      0.261
204835_at        POLA1     -0.482      0.609
```

A summary of gene-specific results for a category is also available from the `safeplot` function. The process of generating SAFE-plots and a more detailed description of the image are given in section 11.1.

```
> safeplot(results, cat.name = "REACTOME:69183", gene.names = genes)
```



5 Building categories from annotation packages

For the example in section 4, REACTOME categories were built internally in `safe` as any term which is annotated to at least two Affymetrix probesets in the filtered dataset (identified by the rownames of `e3.upp`). Categories can also be created externally from `safe`, and stored efficiently as a sparse matrix using the `SparseM` package as follows:

```
> entrez <- unlist(mget(names(genes),hgu133aENTREZID))
> C.mat <- getCmatrix(gene.list = as.list(reactomeEXTID2PATHID),
+                     present.genes = entrez,prefix = "REACTOME:",
+                     min.size = 10, max.size = 500)

1218 categories formed

> results <- safe(e3.upp, p3.upp$p53, C.mat, print.it = FALSE)
```

NOTE: For many instances, when performing pathway-analyses in R, it will improve computational time to create sparse matrices of categories first, and then apply them (or appropriately subsetted objects) to runs of safe with varying expression datasets or response vectors. For instance, we will do so in section 6 to illustrate the different experimental designs that `safe` can accommodate.

Functional categories can also be derived from other sources of information commonly provided in Bioconductor AnnotationData packages. For example, the Protein Families database can be used to generate categories using the argument `annotate = "PFAM"`, or externally from `safe` as:

```
> C.mat2 <- getCmatrix(gene.list = as.list(hgu133aPFAM),
+                      present.genes = rownames(e3.upp))
```

Gene Ontology pathways can also be created from Bioconductor metadata packages. The argument `annotate = "GO.ALL"` will form categories from all three ontologies, while `"GO.CC"`, `"GO.BP"`, or `"GO.MF"` will restrict sets to Cellular Compartment, Biological Process or 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 array features directly annotated to a term, and also to any terms beneath it in the ontology. The C matrix of each can be externally built, under user-defined size restrictions, with the `getCmatrix` function, as follows, :

```
> GO.list <- as.list(hgu133aGO2ALLPROBES)
> C.mat2 <- getCmatrix(keyword.list = GO.list, GO.ont = "CC",
+              present.genes = rownames(e3.upp))
```

Statistical methods for pathway-analyses are generally applicable to any other biological reason for creating a gene-set, but are normally underutilized because of the bioinformatic challenge of creating, storing, and implementing sets. With `safe`, the function `getCmatrix` gives a user the capability, albeit in a somewhat limited fashion, of storing user-defined gene-sets in an ordered manner for analyses. For example, the genes that are measured by the Oncotype DX recurrence score for breast cancer (either all 21 genes, or the 16 non-housekeeper genes), can be tested as a set as:

```
> RS.list <- list(Genes21 = c("ACTB","RPLP0","MYBL2","BIRC5","BAG1","GUSB",
+                             "CD68","BCL2","MMP11","AURKA","GSTM1","ESR1",
+                             "TFRC","PGR","CTSL2","GRB7","ERBB2","MKI67",
+                             "GAPDH","CCNB1","SCUBE2"),
+                 Genes16 = c("MYBL2","BIRC5","BAG1","CD68","BCL2","MMP11",
+                             "AURKA","GSTM1","ESR1","PGR","CTSL2","GRB7",
+                             "ERBB2","MKI67","CCNB1","SCUBE2"))
> RS.list <- lapply(RS.list,function(x)
+                   return(names(genes[which(genes %in% x)])))
> C.mat2 <- getCmatrix(keyword.list = RS.list,
+                      present.genes = rownames(e3.upp))

2 categories formed

> results1 <- safe(e3.upp, p3.upp$er, C.mat2, print.it = FALSE)
> safe.toptable(results1, number = 2, description = FALSE)

  GenesetID Size Statistic P.value Adj.p.value
1   Genes21   51    801431   0.002       0.004
2   Genes16   38    562882   0.012       0.012
```

As shown, the 16- and 21-gene sets in the Oncotype Dx assay are significantly correlated with ER status by IHC, which is expected as one of the gene members ("ESR1" above), and as a widely known prognostic factor for breast cancer. Conversely, no significant association to p53 is seen (data not shown).

Lastly, with version 3.0 of `safe`, a new definition of "soft categories" allows for tests of association to be conducted at the gene-level, instead of the feature-level, of the given platform. This is performed by using an optional argument to `getCmatrix`, `by.gene = TRUE`, and passing the gene annotation as a character vector to `gene.names`. In brief, probesets are downweighted as $\frac{1}{m_g}$, where m_g is the number of probesets annotated to a given gene. As such, the size of the category becomes the number of genes, and the default global statistic becomes a rank-based Wilcoxon linear score statistic. A manuscript is in preparation on "soft categories", and their application to gene-level analysis and other unique biological contexts for pathway-analysis.

6 Experimental designs and local statistics

The two-sample comparison of p53 mutant versus p53 wild type samples is one of several experimental designs that `safe` can automatically accommodate. This section describes the variety of designs, and the corresponding local statistics, along with the arguments in `safe` to execute them. NOTE: To decrease computation time in the following examples, permutation testing is bypassed using the argument `Pi.mat = 1`.

6.1 Two-sample comparisons

For two-sample comparisons, the response vector can either be given as a $(0, 1)$ vector or a character vector with two unique elements. When a character vector is passed to `safe` as the response, the assignment of the first element of the array becomes Group 1, and is printed

as a warning in the output from `gene.results` and `safeplot`. It is critical for the user to be aware of this assignment and how it defines the direction of differential expression.

By default, a Student's t-statistic is employed for categorical comparisons, but if unequal variances are assumed, the Welch t-statistic can be selected using `local = "t.Welch"`. As shown below, feature-by-feature results are highly correlated between the two statistics, as expected.

```
> results2 <- safe(e3.upp, p3.upp$p53, C.mat, local = "t.Welch",
+                  Pi.mat = 1, print.it = FALSE)
> cbind(Student = round(results@local.stat[1:3], 3),
+       Welch   = round(results2@local.stat[1:3], 3))
```

```
          Student Welch
1007_s_at   0.389 0.378
1053_at     1.262 1.254
117_at      0.278 0.296
```

6.2 Multi-class designs

For multi-class designs, response vectors should be character or numeric vectors with unique values for each group. If a character vector with more than two elements 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.

```
> y.vec <- c("p53-/er-","p53-/er+","p53+/er-",
+            "p53+/er+")[1+p3.upp$er+2*p3.upp$p53]
> table(PHENO = y.vec)
```

```
PHENO
p53+/er+ p53+/er- p53-/er+ p53-/er-
      18       13       15        8
```

```
> results2 <- safe(e3.upp, y.vec, C.mat, local = "f.ANOVA",
+                  Pi.mat = 1, print.it = FALSE)
```

6.3 Continuous phenotypes

The example below demonstrates how `safe` can also be used to examine continuous responses, such as tumor size for the breast cancer data set. 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"`.

```
> results2 <- safe(e3.upp, p3.upp$size, C.mat, local = "t.LM",
+                  Pi.mat = 1, print.it = FALSE)
```

6.4 Paired two-sample comparisons

`safe` includes the paired t-test for matched experiments that are $1:1$. To implement 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(1:27,2)*rep(c(-1,1), each = 27)
> results2 <- safe(e3.upp, y.vec, C.mat, local = "t.paired",
+                  Pi.mat = 1, print.it = FALSE)
```

6.5 User-defined local statistics

In addition to these predefined local statistics, `safe` is structured such that the user can specify alternative local statistics by defining a function with the following structure. The primary requirement is that a generic function be loaded which takes as inputs `data`, the matrix of expression data, and `vector`, the response information, as illustrated below. Local statistics should have a null value of 0, whether they are one-sided or two-sided, to be used under default arguments. Additional information can be passed as objects in the optional list, `args.local`. Here, we create a function for a Wilcoxon signed rank, as a non-parametric alternative to the paired t-test described above. NOTE: This choice of local statistic should not be confused with the default global statistic.

```
> local.WilcoxSignRank<-function(X.mat, y.vec, ...){
+ return(function(data, vector = y.vec, ...) {
+   n <- ncol(data)/2
+   x <- data[, vector > 0][, order( vector[vector > 0])]
+   y <- data[, vector < 0][, order(-vector[vector < 0])]
+   ab <- abs(x-y)
+   pm <- sign(x-y)
+   pm.rank <- (pm == 1) * t(apply(ab, 1, rank))
+   return(as.numeric(apply(pm.rank, 1, sum) - n*(n+1)/4))
+ } )
+ }
```

```
> results3 <-  safe(e3.upp, y.vec, C.mat, local = "WilcoxSignRank",
+                   Pi.mat = 1, print.it = FALSE)
> cbind(Paired.t  = round(results2@local.stat[1:3], 3),
+       Sign.Rank = results3@local.stat[1:3])
```

```
          Paired.t Sign.Rank
1007_s_at   -3.372      -132
1053_at      1.267        46
117_at      -0.793       -67
```

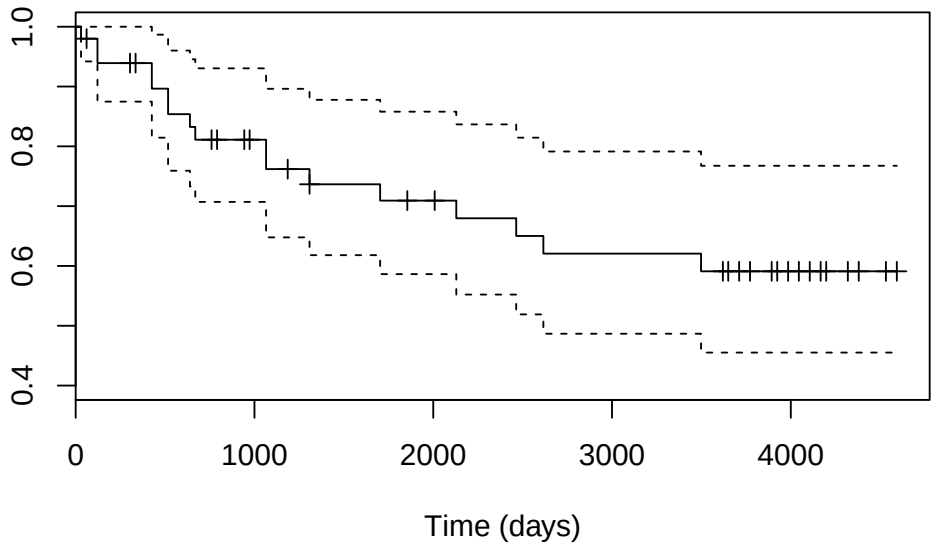
As a resampling-based method, `safe` is computationally intensive, so considerations of efficiency should be made when programming user-defined functions for local and global statistics. The above example, while simple, is much slower than the default paired t-test in `safe` because of its reliance on the `apply` function. Interfacing with C or another foreign language is highly suggested for any statistic without a closed solution that can be written using scalar and matrix operations. Complete instructions on how to design and include user-defined functions will not be included in this vignette.

6.6 New method for survival analysis

A new approach for pathway-analyses with right-censored time-to-event data is provided in `safe` version 3.0, which computes martingale residuals (Therneau *et al.*, 1990) for the

right-censored clinical data, and a score statistic for the association of the continuous residuals to the gene expression estimates. This method is substantially more computationally efficient when compared to using conventional proportional hazards regression models in our resampling-based approaches, and has also been applied to other pathway-analysis tools for gene expression data for this reason (e.g., `globaltest` (Goeman *et al.*, 2005)). The following code illustrates how this approach can be applied to the 17 recurrent and 34 non-recurrent samples in the Grade 3 subset of the breast cancer dataset.

```
> library(survival)

> layout(matrix(1:2,2,1), heights=c(4,2))
> plot(survfit(Surv(p3.upp$t.rfs, p3.upp$e.rfs) ~ 1),
+      xlab = "Time (days)", ylim = c(.4,1))
```




```
> drop <- is.na(p3.upp$t.rfs)
> Xy <- getCOXresiduals(e3.upp[,!drop], p3.upp$t.rfs[!drop],
+                       p3.upp$e.rfs[!drop])
> results2 <- safe(Xy$X.star, Xy$y.star, C.mat,
+                  Pi.mat = 1, print.it = FALSE)
```

6.7 New method for covariate adjustment

Finally, another important extension of the SAFE method allows for resampling-based pathway analysis to be applied to experimental designs with important covariate information (noted here as an $n \times p$ matrix). In principle, defining local statistics in the presence of covariates that can be applied to every array feature is straightforward. However, we still need to handle correlation structures across genes, for which permutation is attractive. The proper handling of covariates is a challenge in the permutation setting, however, as standard permutation forces the investigator to permute the covariates relative to either X or y, but is inappropriate if a covariate is correlated with both X and y. Several permutation approaches in the presence of covariates are described in Good (2000) for linear regression, but are not computationally efficient for high-dimensional datasets. Rather, we propose

computing the residuals X_z and y_z from general or generalized linear regression models for the $n \times p$ covariate matrix Z. Then, the score statistic defined in Zhou *et al.* (2013) can be used as the local statistic in resampling-based tests, or in the analytical approximations using a penalty for the loss of degrees of freedom from adjusting for Z. The following code illustrates how this approach can be applied to the Grade 3 subset of the breast cancer dataset to test for the association of pathways to p53 mutations after adjusting for estrogen receptor status.

```
> table(ER = p3.upp$er, p53 = p3.upp$p53)

   p53
ER   0  1
  0  8 13
  1 15 18

> Xy <- getXYresiduals(e3.upp, p3.upp$p53, Z.mat = p3.upp$er)
> results2 <- safe(Xy$X.star, Xy$y.star, C.mat,
+                  Pi.mat = 1, print.it = FALSE)
```

7 Alternative global statistics

By default, `safe` conducts two-sided tests, taking the absolute value of local statistics, before ranking the feature-by-feature results. In this way, one can identify categories showing either (a) consistent up-regulation, (b) down-regulation, or (c) bi-directional differential expression. An optional argument in `safe` allows users to specify one-sided tests of differential expression: `args.global = list(one.sided = TRUE)` to consider only features in the positive direction to be significant.

In the above SAFE analyses, a functional category was compared to its complementary set of array features with 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 SAFE framework naturally extends to other statistics used in gene category analyses. This allows for one to apply test statistics used in other pathway-analysis software in a way that accounts for gene-gene correlation (see Barry *et al.*, 2008).

7.1 Average difference

Instead of testing the median difference in feature-by-feature association with the Wilcoxon rank sum, a natural analog would be to test the mean difference, as done in *T-profiler* (Boorsma *et al.*, 2005), under an assumption of gene-independence. In `safe`, this can be tested more properly under gene-dependence using the argument: `global = "AveDiff"`, as follows:

```
> results2 <- safe(e3.upp, p3.upp$p53, C.mat,
+                  global = "AveDiff", print.it = FALSE)
> cor(results@global.pval, results2@global.pval, method = "spearman")

[1] 0.964268
```

As shown above, highly concordant results are seen overall between these two choices of global statistics, but the latter will be more sensitive to heavily-tailed empirical distributions of local statistics. This can occur with outliers and highly influential expression estimates that are common in most commercial platforms, despite global transformations (e.g., $\log_2$) to minimize heteroscedasticity among features.

7.2 Gene-list methods

One popular approach to examining categories is through "gene-list enrichment" methods, that were developed as *post hoc* means of inference after the array-features with significant differential expression had been identified. These methods use global statistics that only consider the dichotomous outcomes of feature-by-feature hypothesis tests (i.e., r probesets on an Affymetrix platform are differentially expressed, $m - r$ are not), and typically use Fisher's Exact test or Pearson's test for a difference in proportions. Again, p-values are extremely anti-conservative under the false assumption of gene independence, which can lead to spurious results. For this reason, we have extended `safe` to this class of 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(e3.upp, p3.upp$p53, C.mat, global = "Fisher",
+                  args.global = list(one.sided=F, genelist.length = 200),
+                  Pi.mat = 1, print.it = FALSE)
```

Similarly, the Pearson test for difference in proportions (which is equivalent to a Chi-squared test) can be specified by the argument `global = "Pearson"`, and instead of conditioning on the number of rejected feature-level hypotheses, as in Fisher Exact tests, one specifies the cutoff for the gene-list. This is done in the same manners as shown above, where a one-sided or two-sided threshold value for local statistics is declared by the argument `args.global = list(one.sided = FALSE, genelist.cutoff = 2.0)`.

7.3 Kolmogorov-Smirnov-type tests of enrichment

Lastly, we note that the popular approach to pathway-analysis, Gene Set Enrichment Analysis (GSEA), rightly accounts for gene-gene correlation through permutation-testing (see Barry *et al.*, 2008, for discussion). Rather than using a global statistic for comparing central tendencies between a category and its complement, Subramanian (2005) proposed a Kolmorgorov-Smirnov statistic used traditionally as a non-parametric test for more general differences in distributions.

In `safe`, this statistic can be used with the argument `global = "Kolmogorov"`, although we note that this will be more computationally intensive than the above options, since it cannot rely solely on scalar and matrix operators for calculation. For this reason, it is not applied in this vignette and may not be feasible for general use, other than for simulation studies to compare against output from GSEA and other softwares.

8 Adjusting for multiple comparisons in SAFE

As in standard feature-by-feature analyses, it is necessary to account for multiple comparisons when considering a set of categories. By default, `safe` accounts for multiple comparisons by reporting the Benjamini-Hochberg (1995) estimate of the *false discovery rate* (FDR), `error = "FDR.BH"`, with every nominal p-value. `safe` also includes options for Bonferroni correction, `error = "FWER.Bonf"` or Holm's step-down procedure, `error = "FWER.Holm"`, for the family-wise error rate (FWER).

Since SAFE is a resampling-based test, permutation-based error rate methods have been incorporated into `safe` which will control the correlation between tests of categories with overlapping or non-overlapping but co-regulated genes. This includes Yekutieli-Benjamini (1999) estimates of the FDR, `error = "FDR.YB"`, and the Westfall-Young (1989) method, `error = "FWER.WY"`, for controlling the FWER. Although we feel these two permutation-based procedures for controlling error are superior (by empirically accounting for correlation among tests), they are more computationally and memory intensive, requiring all permuted global statistics be stored from resampling. For this reason, we have not included them in the vignette, and the traditional Benjamini-Hochberg (1995) estimate is selected by default.

9 Bootstrap-based tests in SAFE

In Barry *et al.* (2008), a bootstrap-based version of SAFE was proposed and shown to generally be more powerful while controlling Type I error. Two basic methods of hypothesis testing defined by Efron (1982) 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 computed from the resampled 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 example is an anecdotal illustration of increased power from bootstrap-resampling; a more definitive demonstration using simulation appears in Barry *et al.* (2008).

```
> set.seed(12345)
> results2 <- safe(e3.upp, p3.upp$er, C.mat2,
+                  method = "bootstrap.t", print.it = FALSE)
> round(cbind(Perm = results1@global.pval,
+             Boot.t = results2@global.pval),4)
```

```
         Perm Boot.t
Genes21 0.002  3e-04
Genes16 0.012  7e-04
```

Based on the requirements for bootstrap-based hypothesis testing (see Barry *et al.*, 2008), they can only be performed using 1) Wilcoxon rank sum, 2) average difference, or 3) Pearson difference in proportions (the pre-defined global statistics).

P-values for local statistics are calculated under bootstrap resampling, using 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 must relate to no differential expression in the supplied local statistics.

By default, the data are resampled $B = 1000$ times when performing permutation or quantile bootstrap tests. However, with minimum empirical p-values of $\frac{1}{B}$, this may not be

sufficient when testing hundreds or thousands of categories, and $\geq$ 10-fold more resamples could be needed if there are several hundred categories being investigated. The Gaussian bootstrap-based test has the advantage that empirical p-values are not bounded by the total number of resamples taken, such that small p-values can be obtained without intensive computational effort.

Moreover, because permutation-resampling is intensive in computation time and memory requirements, we have recently developed accurate analytic approximations to permutations of score statistics that have good performance for even relatively small sample sizes (Zhou *et al.*, 2013). This approach preserves the essence of controlling Type I error by permutation pathway analysis, but with greatly reduced computation, and is more accurate than competing moment-based approaches in common use. The method has improved properties, in terms of mean square error, over the common use of random sampling to select permutations. These algorithms have been written into a new R package, `safeExpress`, which can be called internally by `safe` using the argument `method = express`, so that the output is provided as a *SAFE* object. The `safeExpress` package is available by request to any of the vignette coauthors.

10 Parallel processing

The default method in `safe` calculates p-values empirically by resampling. While standard `R` conducts each operation sequentially, under parallel processing multiple computational tasks are executed simultaneously on separate computer processing cores. A new feature in `safe` allows users to take advantage of the multiple computing cores commonly available on servers and PCs to get substantial improvements in computing time by conducting bootstrap or permutation resampling in parallel. Similar improvements are seen with numerical approximations in `method = "express"` by testing individual categories in parallel.

10.1 Implementation

When executing `safe` with any available method (permutation, bootstrapping, or "express"), with `error = "none", "FWER.Bonf",` or `"FDR.BH"`, users can specify the optional parameter `parallel = TRUE` to leverage parallel processing. Within `safe`, the `foreach` package is implemented for parallel execution of the requested method for the primary analysis. The `foreach` package provides a "frontend" for any available parallel "backend" initialized prior to calling the `safe` function. The `foreach` package is compatible with a variety of backends supporting `%dopar%` functionality, and has been tested successfully with `doMC, doMPI, doParallel`, and `doSNOW`. An example invocation is given below, but users should consult the documentation of these packages for more information on how to initialize a parallel backend prior to invoking `safe` with `parallel = TRUE`. Upon execution, if no such backend is available, the analysis will proceed in sequence, so conditional coding is not required, ensuring code portability. When running in parallel, `safe` uses the `doRNG` package to allow users to set seeds for random number generation, enabling reproducible analyses.

```
> #Initialize parallel backend
> library(doParallel)
> registerDoParallel(cores=4)
> set.seed(12345)
```

n	m	C	Sequential Permutation	Parallel Permutation	Relative	Sequential Bootstrap	Parallel Bootstrap	Relative
S	S	S	81.5	21.0	3.9	17.8	5.5	3.2
S	L	S	265.8	59.9	4.4	44.0	15.2	2.9
L	S	S	319.5	79.1	4.0	62.9	16.8	3.7
S	S	L	330.0	75.7	4.4	68.3	17.1	4.0
L	S	L	418.4	126.0	3.3	127.2	26.5	4.8
S	L	L	581.4	149.7	3.9	125.7	33.1	3.8
L	L	S	1380.2	343.9	4.0	246.5	65.7	3.8
L	L	L	1476.6	368.3	4.0	246.3	74.1	3.3

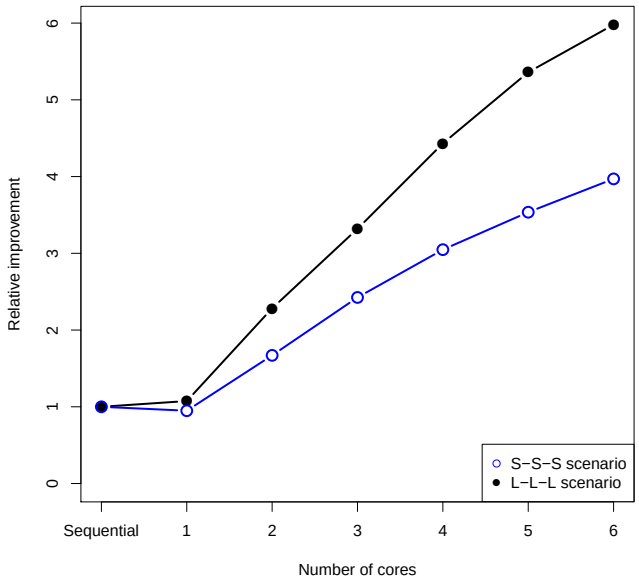
Table 1: Average run time (seconds) over 10 executions. "Relative" columns give sequential execution time divided by parallel execution time. "S" and "L" indicate small or large testing scenario parameters.

```
> results <- safe(e3.upp, p3.upp$p53, platform = "hgu133a.db",
+                 annotate = "REACTOME", print.it = FALSE, parallel=TRUE)
```

10.2 Comparison of methods

To demonstrate the efficiency gains possible under parallel processing, a comparison study was conducted using the Miller *et al.* (2005) data to compare performance in a variety of scenarios. To illustrate the breadth of conditions under which SAFE may be implemented, "large" and "small" numbers of samples, probesets, and categories were combined, forming eight different scenarios for testing. The large sample size uses all of the phenotype data (251 samples), while the small sample size is limited to only those samples indicated as Grade 3 (54 samples). The large probeset scenarios use all probesets from the complete Affymetrix hgu133a array (44928 probesets), while the small scenarios drop the Affymetrix control probesets and truncate the data to one probeset per gene (12702 probesets). Finally, the large and small sets of categories are constructed using the Gene Ontology pathways (12140 or 13873 categories) or the Protein Families database (2061 or 2764 categories), respectively, depending on the number of probesets being used.

Sequential and parallel analyses using the permutation method used the default setting of 1000 permutations, and bootstrapping methods used the default of 200 bootstrap resamplings. The analyses for each testing scenario was repeated 10 times, and the resulting average processing times are summarized in table 1. Table 1 was generated using 4 parallel processing cores on a 64-bit Debian server with two dual-core AMD Opteron processors (four cores total) with 16GB of RAM, running `R` version 2.15.2. As expected, parallel executions of the permutation method were 4-4.5 times faster than sequential executions, and parallel executions of the bootstrapping method were 3-5 times faster.



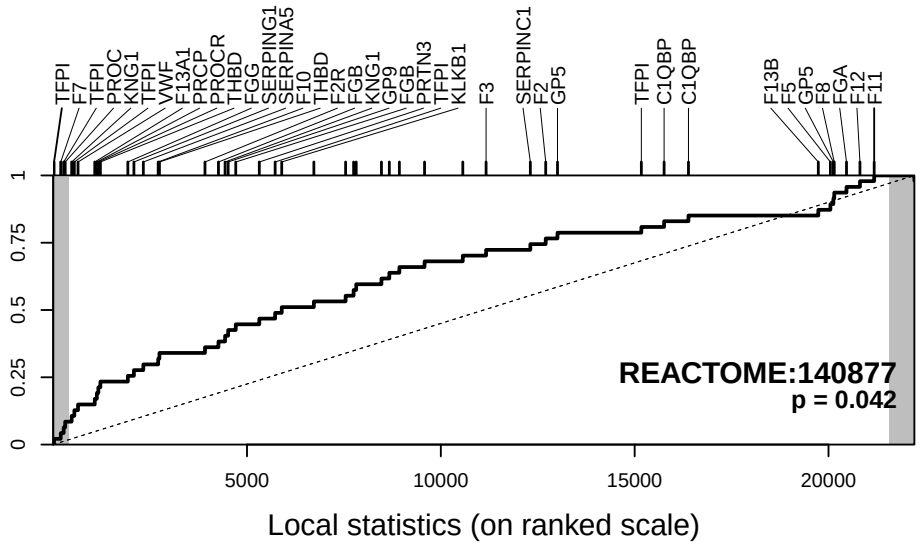
Because computational overhead is minimal outside of the resampling loop, executions times increase linearly with the number of resamplings but decrease linearly with the number of cores. The analyses for the permutation method for two testing scenarios were repeated 10 times each using an increasing number of parallel processing cores on a 64-bit Debian server with a six-core Intel i7-3960X Extreme processor with 48GB of RAM, running `R` version 2.15.2. The resulting processing times, relative to the sequential execution time, are summarized in the figure above. The S-S-S and L-L-L scenarios correspond to those described for table 1. As the number of samples, probesets, or categories in the data is increased, so is the computational burden of the resampling loop relative to that outside the loop. Thus parallel processing shows more substantial improvement relative to sequential processing for larger data sets.

11 Visualizing pathway-level association in SAFE

11.1 SAFE-plots

Ever since permutation testing for pathway-analysis was used in Virtaneva *et al.* (2001), we have advocated that the cumulative distribution function (CDF) of the category be used to visualize the relative magnitude and direction of differential expression of array features annotated to the category. By default, the function `safeplot` will create a figure for the most significant pathway, as shown in section 4. SAFE-plots of other categories can be generated with the argument `cat.name`.

```
> safeplot(results, cat.name = "REACTOME:140877", gene.names = genes)
```



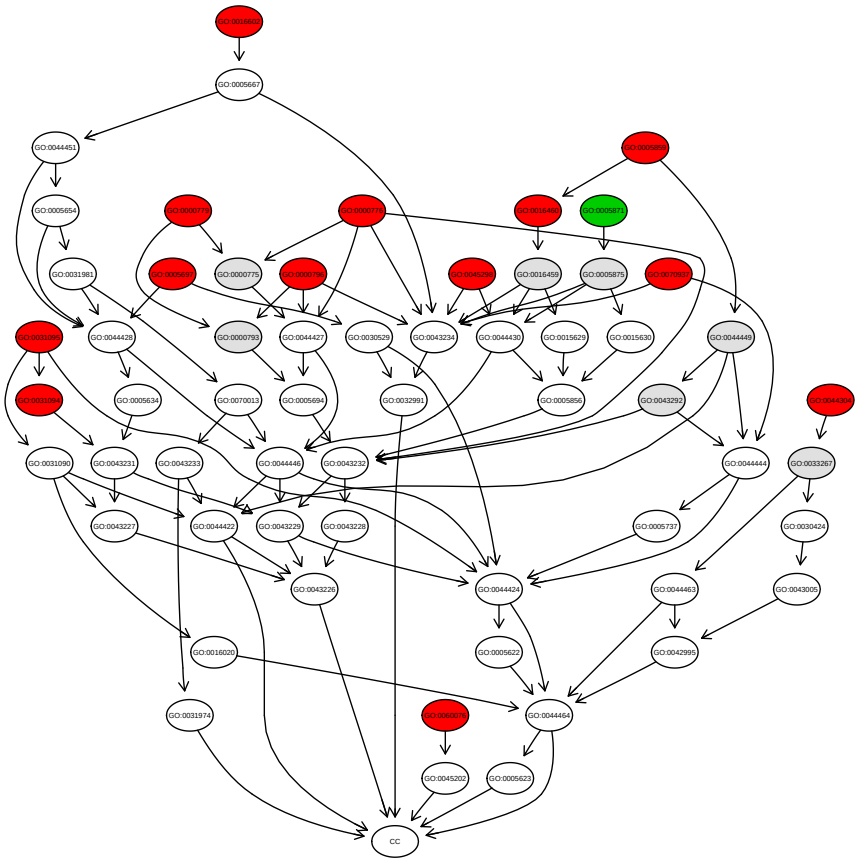
SAFE-plots show the empirical CDF for the local statistics from a given category (solid line), on the rank scale by default, or on an unranked scale with argument `rank = FALSE`. A significant category will have more extreme associations to the response of interest than its complement, resulting in either a right-ward, left-ward, or bidirectional shift in the CDF away from the overall CDF (dashed line, which is uniform on the ranked scale). The shaded regions of the plot correspond to the features that pass a nominal level of significance (empirical p-values ≤ 0.05 by default). The features in the category are shown as tick marks along the top of the graph, and depending on the category size, either all features in the category are labeled, or only the most extreme ones that will fit in the negative and positive areas of the plot.

This SAFE-plot shows that the features of KEGG pathway 03030 demonstrate consistent overexpression in p53+ samples versus p53−, including PRIM2, MCM*, and RFC* genes reaching a nominal level of significance individually. In contrast, KEGG:00072 (above) does not show a consistent pattern of differential expression, with only a single strongly significant gene, ACT2, likely driving the association to p53 status.

11.2 Directed acyclic graphs of gene categories

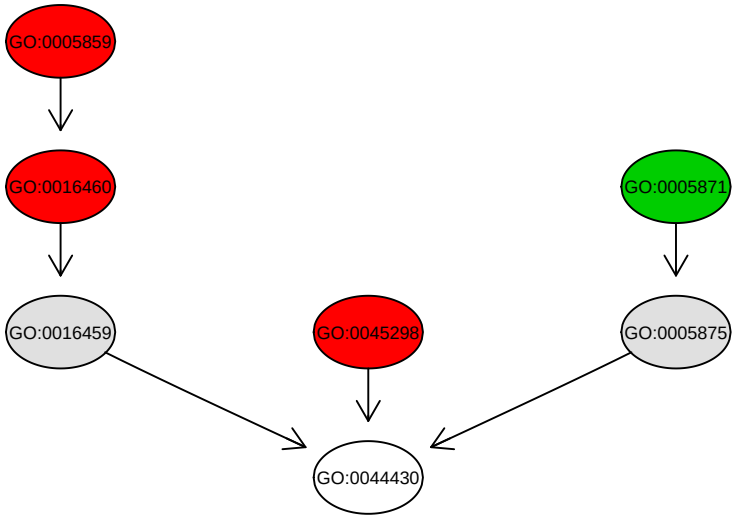
Finally, Gene Ontology is a unique, structured vocabulary where genes are annotated from broad to narrow levels of classification in a directed acyclic graph (DAG). As such, many categories are highly related in their gene membership, and visualizing results across the ontology can be useful in ascertaining 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`.

```
> 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:0044430")
```



12 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. (2008) 'A Statistical Framework for Testing Functional Categories in Microarray Data', *Annals of Applied Statistics*, **2**(1), 286–315.
- Benjamini Y., Hochberg Y. (1995) 'Controlling the False Discovery Rate - a Practical and Powerful Approach to Multiple Testing', *Journal of the Royal Statistical Society Series B-Methodological*, **57**, 289–300.
- Boorsma, A., Foat, B.C., Vis, D. Klis, F., and Bussemaker, H.J. (2005) 'T-profiler: scoring the activity of predefined groups of genes using gene expression data', *Nucleic Acids Res*, **33**, 592–595.
- Edgar R., Domrachev M., and Lash A.E (2002) 'Gene Expression Omnibus: NCBI gene expression and hybridization array data repository', *Nucleic Acids Res.*, **30**(1), 207–210.
- Efron, B. (1982) *The jackknife, the bootstrap, and other resampling plans*, Philadelphia, PA: Society for Industrial and Applied Mathematics.
- Gatti, D.M., Barry, W.T., Nobel. A.B., Rusyn, I., and Wright, F.A., (2010) 'Heading Down the Wrong Pathway: on the Influence of Correlation within Gene Sets', *BMC Genomics*, **11**, 574.
- Goeman, J.J., Oosting, J., Cleton-Jansen, A.M., Anninga, J.K. and van Houwelingen, H.C. (2005) 'Testing association of a pathway with survival using gene expression data', *Bioinformatics*, **21** (9), 1950–1957.
- Good, P.I. (2000) *Permutation tests : a practical guide to resampling methods for testing hypotheses*, New York, NY: Springer.
- Miller, L.D., Smeds, J., George, J., Vega, V.B., Vergara, L., Ploner, A., Pawitan, Y., Hall, P., Klaar, S., Liu, E.T., and Bergh, J. (2005) 'An expression signature for p53 status in human breast cancer predicts mutation status, transcriptional effects, and patient survival', *Proc Natl Acad Sci U S A*, **102**(38), 13550–13555.
- Subramanian, A., Tamayo, P., Mootha, V.K., Mukherjee, S., Ebert, B.L., Gillette, M.A., Paulovich, A., Pomeroy, S.L., Golub, T.R., Lander, E.S., and Mesirov, J.P. (2005) 'Gene set enrichment analysis: A knowledge-based approach for interpreting genome-wide expression profiles', *Proc Natl Acad Sci U S A*, **102**(43), 15545–15550.
- Therneau, T.M., Grambsch, P.M., and Fleming, T.R. (1990) 'Martingale-based residuals for survival models', *Biometrika*, **77**(1), 147–160.
- Virtaneva, K.I., Wright, F.A., Tanner, S.M., Yuan, B., Lemon, W.J., Caligiuri, M.A., Bloomfield, C.D., de laChapelle, A. and 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.

- 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.
- 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.
- Zhou, Y.H., Barry, W.T., and Wright, F.A. (2012) 'Empirical Pathway Analysis, without Permutation', *Biostatistics*, **14**(3):573–85.