Class Comparison with OOMPA

Kevin R. Coombes

September 5, 2024

Contents

1 Introduction

OOMPA is a suite of object-oriented tools for processing and analyzing large biological data sets, such as those arising from mRNA expression microarrays or mass spectrometry proteomics. The *ClassComparison* package in OOMPA provides tools to work on the "class comparison" problem. Class comparison is one of the three primary types of applications of microarrays described by Richard Simon and colleagues. The point of these problems is to identify genes that behave differently in known classes; in other words, a typical class comparison problem is to find the genes that are differentially expressed between two types of samples.

2 Getting Started

No one will be surprised to learn that we start by loading the package into the current R session:

```
> library(ClassComparison)
```

The main functions and classes in the ClassComparison package work either with data matrices or with `ExpressionSet` objects from the BioConductor *Biobase* package. For the first set of examples in this vignette, we will use simulated data that represents different groups of samples:

```
> set.seed(6781252) # for reproducibility
> nGenes <- 5000
> nSamp <- 15
> nDif <- 150
> delta <- 1
> fake.class <- factor(rep(c('A', 'B'), each=nSamp))
> fake.data <- matrix(rnorm(nGenes*nSamp*2), nrow=nGenes, ncol=2*nSamp)
> fake.data[1:nDif, 1:nSamp] <- fake.data[1:nDif, 1:nSamp] + delta
> fake.data[(nDif+1):(2*nDif), 1:nSamp] <- fake.data[(nDif+1):(2*nDif),
+                                                    1:nSamp] - delta
```

3 Gene-by-gene t-tests

The simplest way to find differentially expressed genes is to perform a two-sample t-test on each gene. The `MultiTtest` class handles this operation, with a summary that carefully ensures that you know which class is associated with a positive t-statistic.

```
> mtt <- MultiTtest(fake.data, fake.class)
> summary(mtt)
```

```
Row-by-row two-sample t-tests with 5000 rows
Positive sign indicates an increase in class: A

Call: MultiTtest(data = fake.data, classes = fake.class)

T-statistics:
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
-5.703648 -0.732668  0.002954  0.008923  0.737256  7.021149

P-values:
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
0.0000001 0.2091023 0.4680792 0.4755566 0.7430658 0.9999677
```

4 Beta-uniform mixture models to account for multiple testing

As everyone now knows, an inherent difficulty with performing a separate test for each gene is that the p-values must be adjusted to account for multiple

```
> hist(mtt, breaks=101)
```

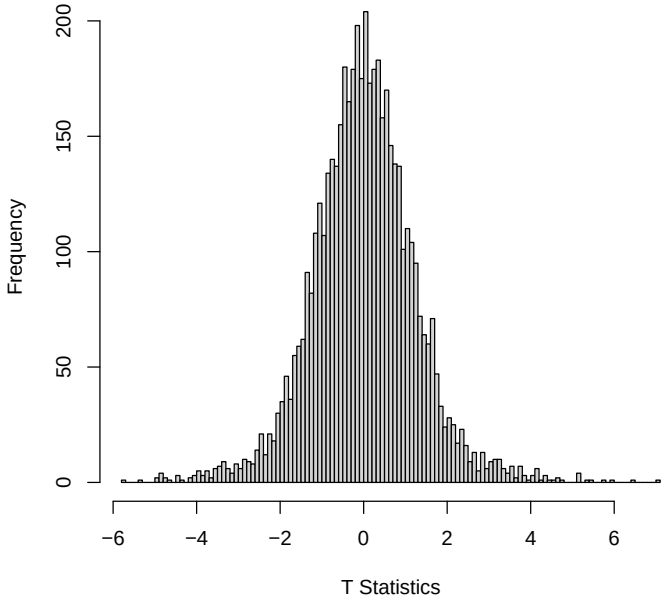

Figure 1: Histogram of the gene-by-gene two-sample t-statistics

testing. A simple approach models the set of p-values using a beta-uniform mixture (BUM). We can perform this analysis with a single command:

```
> bum <- Bum(mtt@p.values)
> summary(bum)

Beta-Uniform Mixture Model

MLE Estimates: ahat = 0.33804 , lhat = 0.87159
Upper Bound on Fraction Unchanged: pihat = 0.915

   tau          TP          FN           FP        TN
1 0.01 0.02663749 0.05836417 0.009149983 0.9058484
```

The default value of the `summary` command is not very enlightening, but we can get a graphical overview of the distribution. The region below the horizontal blue line in Figure 2 represents the uniform component of the mixture (i.e., genes that are not differentially expressed); the region between the blue line and the green curve represents the beta component (i.e., genes that are differentially expressed). If we set a threshold for significance using some cutoff on the p-value (such as the one indicated by the vertical purple line in Figure 2), then we can divide the area into four regions representing true positives, false positives, true negatives, and false negatives. These areas can then be used to estimate the false discovery rate (FDR) as a function of the threshold (Figure 3).

The usual application of this idea is to choose a threshold that achieves a desired level of FDR. For example, selecting genes with a p-value less than

```
> cutoffSignificant(bum, alpha=0.10, by="FDR")

[1] 0.001847872
```

should keep the FDR less than 10%. The number of such genes is easily obtained with the command:

```
> countSignificant(bum, alpha=0.10, by="FDR")

[1] 98
```

You can also get a logical vector that selects the significant genes:

```
> selected <- selectSignificant(bum, alpha=0.10, by="FDR")
```

In our example, the truly significant genes are among the first 300 genes. We can use this information to find out how close we are to the truth; the achieved FDR in this simulated example is pretty close to the target value of 10%.

```
> truth <- rep(FALSE, nGenes)
> truth[1:(2*nDif)] <- TRUE
> sum(selected & truth)

[1] 91

> mean(!truth[selected])

[1] 0.07142857
```

```
> hist(bum)
> abline(v=0.05, col="purple", lwd=2)
```

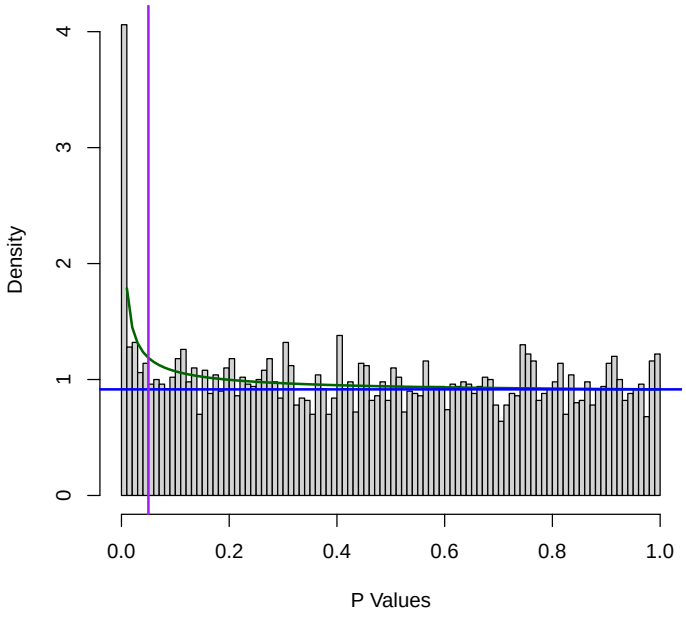

Figure 2: Results of the BUM analysis of the p-values.

```
> image(bum)
>
```

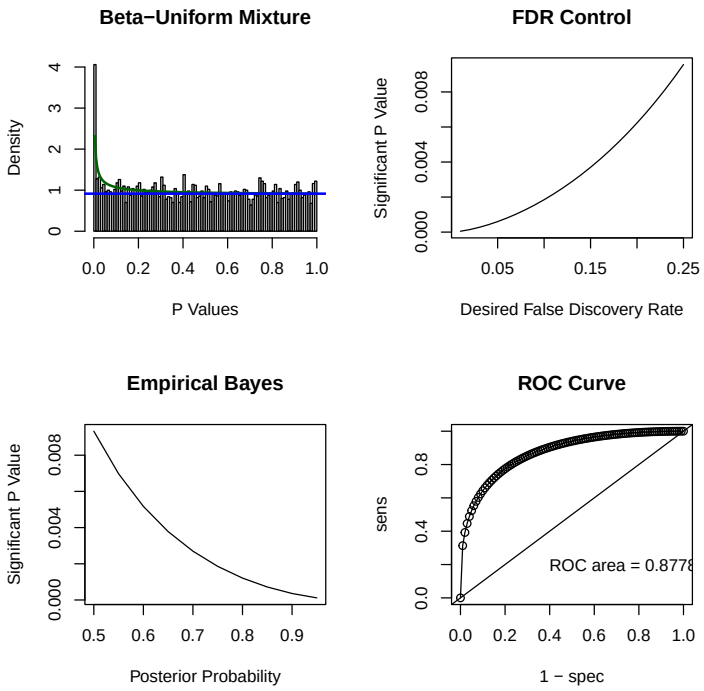

Figure 3: Results of the BUM analysis of the p-values.

5 Wilcoxon rank sum tests and empirical Bayes

In many applications of microarrays, it is unclear how the data should be transformed to achieve the approximate normality needed to justify a t-test. It may just be simpler to ignore the transformation problem and use nonparametric methods, like the Wilcoxon rank-sum test, that only use the ranks of the samples for the expression of each gene.

```
> mw <- MultiWilcoxonTest(fake.data, fake.class)
> summary(mw)

Call: MultiWilcoxonTest(data = fake.data, classes = fake.class)
Row-by-row Wilcoxon rank-sum tests with 5000 rows

Rank-sum statistics:
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  130.0   215.0   233.0   232.7   250.0   338.0

Large values indicate an increase in class: A

With prior = 1 and alpha = 0.9
        the upper tail contains 23 values above 313
        the lower tail contains 18 values below 152
```

A histogram (Figure 4) of the Wilcoxon statistics indicates that the observed values have larger tails than expected by chance, suggesting that we ought to be able to pick out some genes that are significantly different. To do this, we use an empirical Bayes method originally suggested by Efron and Tibshirani. The idea is that we can decompose the Wilcoxon statistics as a mixture of those that arise from the null distribution (which is Wilcoxon with parameters based on the number of samples in each group) and some other component representing the differentially expressed genes. In that case, we can write the observed distribution $f(x)$ in the form:

$$f(x) = \pi f_0(x) + (1 - \pi) f_1(x)$$

where $f_0(x)$ is the known Wilcoxon distribution and $f_1(x)$ is unknown. Since we can estimate $f(x)$ from the observed data, we can simply solve for the unknown distribution $f_1(x)$ provided we know the mixing parameter π, which represents the prior probability that a gene is not differentially expressed. The "empirical" part of this empirical Bayes method comes down to selecting the prior π after looking at the data. For, if we start with $\pi = 1$, the posterior probability of being differentially expressed as a function of the observed statistic ends up taking on negative values (Figure 5), which is rather unpleasant.

By trial and error, we can find a value for π that ensures that the posterior probabilities are always positive (Figure 6). In this case, something close to 0.94 works okay. We can then use a threshold on the posterior probabilities to set a significance cutoff on the Wilcoxon statistics.

```
> hist(mw)
```

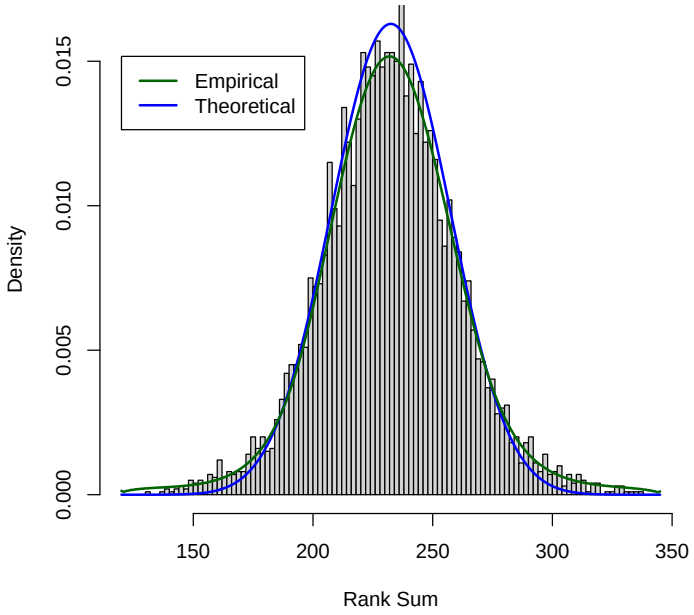

Figure 4: Histogram of the observed gene-by-gene Wilcoxon statistics.

```
> plot(mw)
> abline(h=0)
```

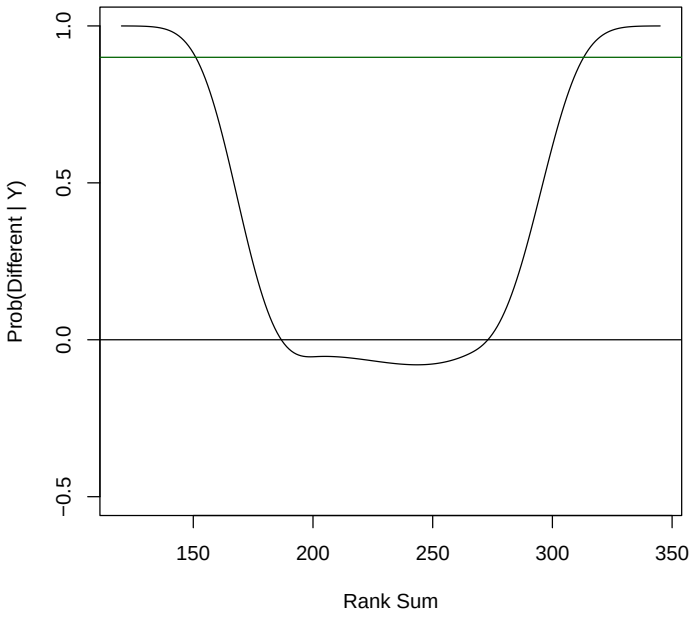

Figure 5: Plot of the posterior probability of being differentially expressed, assuming *a priori* that no genes are different.

```
> plot(mw, prior=0.92, signif=0.9)
> abline(h=0)
```

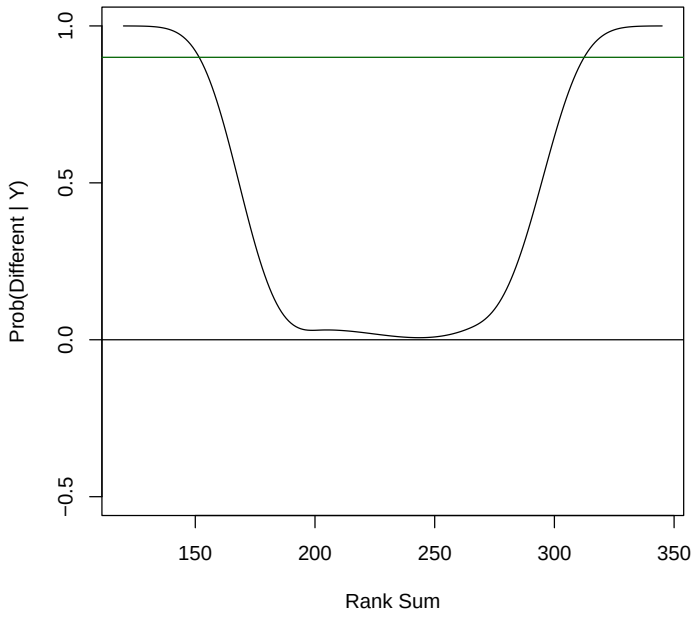

Figure 6: Plot of the posterior probability of being differentially expressed, assuming *a priori* that 92% of the genes are not different.

```
> cutoffSignificant(mw, prior=0.94, signif=0.8)

$low
[1] 158

$high
[1] 306

> countSignificant(mw, prior=0.94, signif=0.8)

[1] 80

> wilsel <- selectSignificant(mw, prior=0.94, signif=0.8)
> sum(selected & wilsel)

[1] 76

> sum(truth & wilsel)

[1] 73
```

6 Permutation based methods

The `Bum` method, as applied to the gene-by-gene t-tests of the *MultiTtest* class or to p-values fmo other tests, assumes that genes are independent. This assumption is clearly false, and so various researchers have proposed permutation-based methods that retain the correlation structure between genes when trying to estimate the distribution of p-values.

6.1 Dudoit's method based on Westfall and Young

Sandrine Dudoit and colleagues introduced the idea of using the Westfall-Young stepdown procedure to control the family-wise error rate in a microarray study. In our example, we can perform this analysis as follows:

```
> dudoit <- Dudoit(fake.data, fake.class, nPerm=100)

1.2.3.4.5.6.7.8.9.10.11.12.13.14.15.16.17.18.19.20.21.22.23.24.25.26.27.28.29.30.31.32.33.34

> summary(dudoit)

Row-by-row two-sample t-tests with 5000 rows
Positive sign indicates an increase in class: A

Call: Dudoit(data = fake.data, classes = fake.class, nPerm = 100)

T-statistics:
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
```

```
> plot(dudoit)
```

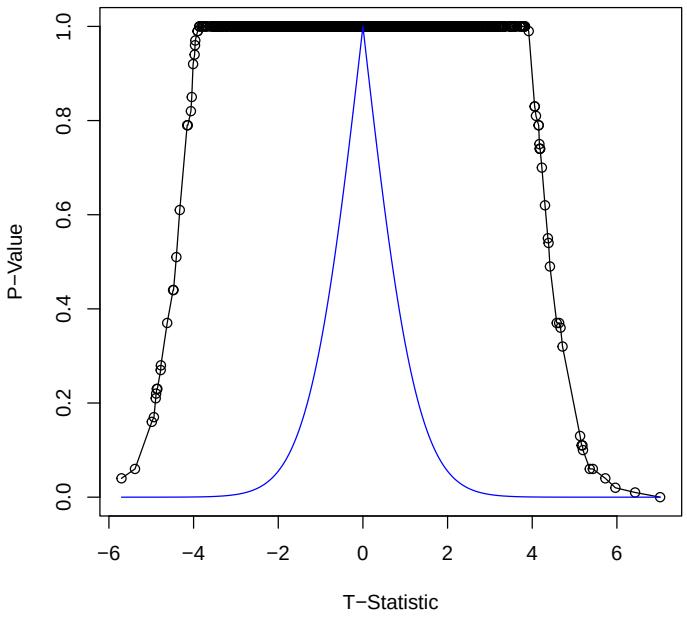

Figure 7: Plot of the unadjusted (blue) and adjusted (black) p-values.

```
-5.703648 -0.732668  0.002954  0.008923  0.737256  7.021149

P-values:
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
0.0000001 0.2091023 0.4680792 0.4755566 0.7430658 0.9999677
```

To get good results, we probably need more than 100 permutations, but this implementation (completely in R) is rather slow. The default plot routine (Figure 7) shows both the unadjusted and adjusted p-values. In most cases, controlling the family-wise error rate (FWER) is viewed as overly conservative, since it tries to ensure that there are no false positive findings instead of trying to estimate the number or fraction of false positives. In our example, using the Dudoit correction with FWER = 10% finds very few differentially expressed genes:

```
> countSignificant(dudoit, 0.10)

[1] 8
```

```
> plot(sam, tracks=seq(0.5, 2, by=0.5))
```

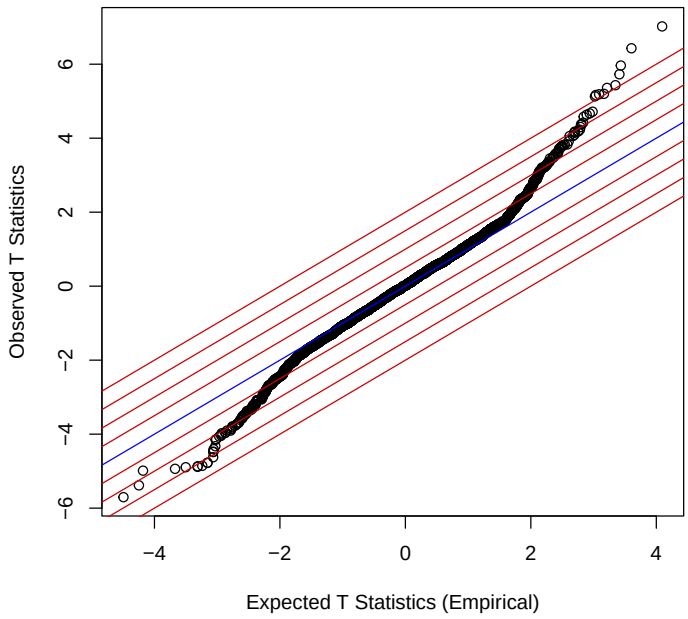

Figure 8: Quantile-quantile plot of the observed t-statistics against the t-statistics expected from the permutation-based null distribution.

7 Significance Analysis of Microarrays

Significance Analysis of Microarrays (SAM) is an alternative procedure, which is based on permutations but tries to control the FDR instead of the FWER. We can get the results by:

```
> sam <- Sam(fake.data, fake.class)

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

> summary(sam)

Using a cutoff of 1 , we called 102 genes significant with expected FDR = 0.0882  ( 9 )
```

Based on the figure, we can probably take a cutoff of 1 to define significance, which yields the following results

```
> cutoff <- 1
> countSignificant(sam, cutoff)
```

```
[1] 102
```

```
> sum(selectSignificant(sam, cutoff) & truth)
```

```
[1] 94
```

8 Other class comparison approaches

The package contains several other methods for finding genes that are differentially expressed between known classes:

1. Total Number of Misclassification (`TNoM`): This method was introduced by Yakhini and Ben-Dor and applied by Bittner and colleagues in 2000. It has probably seen fewer applications than it deserves, which this implementation may help rectify.

2. The smooth (regularized) t-test (`SmoothTtest`): This method was introduced by Baggerly and Coombes in 2001, but a large number of authors have proposed similar ideas. The basic idea is that (even after log transformation) genes of similar intensity appear to have similar variance, and that one can borrow strength across genes to get better estimates of the variability even in small microarray studies.

3. Linear models (`MultiLinearModel`) can be constructed using the usual R formula, providing generalization to one-way designs with more than two classes to compare, or to factorial designs.

4. Gene-by-gene paired t-tests (`MultiTtestPaired`).

5. Gene-by-gene t-tests assuming unequal variance in the two groups (`MultiTtestUnequal`).

A future version of this vignette may include more examples of their use; for now, you can read the examples in the help pages.