User's Guide to BUM Library version 1-1

Stan Pounds

January 8, 2003

Legal Notice

Any damages arising from use of this software or any of its components are NOT the responsibility of its developer, Stan Pounds or his employer, St. Jude Children's Research Hospital.

Introduction

The BUM library implements procedures designed to help in estimating and controlling the occurrence of false positives and false negatives in microarray analysis. The method and requirements for its proper use are described in Pounds and Morris (2003).

1 Getting Started

Download the library from XXXXXX and use the S-plus command *source* to make the functions in the library available. The library is thoroughly documented. The purpose, required arguments, and returned values for each function are more thoroughly explained in the code file in comments marked by #.

2 Obtaining p-values

It is strongly recommended that no filtering be used prior to application of this method. The method cannot estimate the errors made in any prior filtering. An appropriate statistical technique should be used to obtain a p-value for testing the hypothesis of interest for each gene or probe individually. The p-values should be stored in an S-plus vector. In the following discussion, it will be assumed that the p-values are stored in a vector named *pvals*, though the user can store them under a different name.

3 Obtaining Maximum Likelihood Estimates

Once the p-values have been obtained, one should use maximum likelihood estimation (MLE) to find the values of the BUM parameters that correspond to the BUM model that best describes the observed p-values. The command

```
MLE<-bum.mle(pvals,nstartpts,starta,startlambda,nadj,epsadj)
```

will compute the MLE's for the parameters a and λ of the BUM model and store them in an object called *MLE*. The arguments *nstartpts*, *starta*, *startlambda*, *nadj*, and *epsadj* are optional arguments of *bum.mle* that handle the details of starting values and adjusting p-values if some p-values are equal zero. For more information on these arguments, see the documentation in the S-plus library by finding the string "Function: bum.mle" in the S-plus library code file. The object *MLE* contains information about the maximum likelihood estimate.

4 Checking the Adequacy of the Fitted Model

The adequacy of the fitted BUM model should be checked with a quantile-quantile plot. The quantile-quantile plot should be approximately linear if the model is adequate. The command

```
qqbum(pvals,a=MLE$a,lambda=MLE$lambda,optional_arguments)
```

will produce a quantile-quantile plot. There are optional arguments for *qqbum* that allow the user to specify the title and axis labels. If the arguments *a* and *lambda* are not provided, *qqbum* will call *bum.mle* to estimate them. The other optional arguments for *qqbum* are the optional arguments to provide to *bum.mle* when *a* and *lambda* are not specified. DO NOT CONTINUE THE ANALYSIS IF THE QUANTILE-QUANTILE PLOT INDICATES SUBSTANTIAL DEVIATION FROM THE FITTED MODEL. Another useful way to assess the adequacy of the model is to compare the fitted model to a histogram. The command

```
bum.histogram(pvals,a=MLE$a,lambda=MLE$lambda,optional_arguments)
```

will produce a histogram and overlay the fitted BUM curve. To learn more about *qqbum* or *bum.histogram* see the documentation by searching in the code file for the strings "Function: qqbum" or "Function: bum.histogram" respectively.

5 Estimating Error Occurrence Quantities

If a threshold of significance *tau* is specified, a variety of error occurrence quantities can be estimated. The command

```
bum.FDR(tau,a=MLE$a,lambda=MLE$lambda)
```

will compuate an estimated upper bound for the false discovery rate (FDR) of Benjamini and Hochberg (1995). The command

```
bum.EBP(tau,a=MLE$a,lambda=MLE$lambda)
```

will compute an estimated lower bound for the empirical Bayes' posterior (EBP) proposed by Efron, et al. (2001). The command

```
bum.false.positive(tau,a=MLE$a,lambda=MLE$lambda)
```

will compute an estimated upper bound for the proportion of all tests resulting in a false positive (Type I error). The command

```
bum.false.negative(tau,a=MLE$a,lambda=MLE$lambda)
```

will compute an estimated lower bound for the proportion of all tests resulting in a false negative (Type II error). Additionally, the command

```
ext.pi(a=MLE$a,lambda=MLE$lambda)
```

will compute an estimated upper bound for the proportion of genes or probes following the null hypothesis.

6 Controlling Error Occurrence Quantities

Alternatively, one can select the threshold of significance to control selected error occurrence quantities. To find a threshold for the p-value that sets the estimated bounds for the FDR or EBP equal to *gamma* use

```
find.FDR.threshold(gamma,a=MLE$a,lambda=MLE$lambda)
```

or

```
find.EBP.threshold(gamma,a=MLE$a,lambda=MLE$lambda)
```

respectively.

7 Accounting for the Variability of Estimates

The function *bum.CI* can be used to find approximate confidence intervals for many quantities estimated in the analysis. All confidence intervals computed by *bum.CI* depend upon a set of confidence contours determined by *bum.logL.contour*. The accuracy of the intervals computed by *bum.CI* depends upon the accuracy of the contour object supplied to it. So first, one must find the confidence contours. The command

```
cntrs<-bum.logL.contour(pvals,conflevel=0.95,optional_arguments)
```

will find an estimated 95% confidence region (or contour) for the parameters a and λ and store the results in an S-plus object named *cntrs*. The optional arguments of *bum.logL.contour* control options that affect the accuracy of the contours, computing time, and control whether or not the contours are plotted. To learn more about the optional arguments of *bum.logL.contour* search for the string "Function: bum.logL.contour" in the S-plus library file. To obtain confidence intervals with different levels, supply a vector to the argument *conflevel*.

The estimated confidence contour stored in *cntrs* forms the basis of finding 95% confidence intervals for other quantities. The function *bum.CI* has four arguments: *bc.object*, *quantity*, *arg* and *MLE*. The argument *bc.object* should contain the object returned by *bum.logL.contour*. The argument *quantity* is a string specifying the quantity of interest. The argument *arg* gives the value of an additional argument needed to compute the confidence interval. The argument *MLE* is optional, but if provided it should contain the results of *bum.mle*. If *MLE* is specified, *bum.CI* will return a point estimate for the quantity as well. As an example, suppose one wants to find a confidence interval for the false discovery rate if significance is declared by comparing p-values to 0.01. The command

```
bum.CI(bc.object=cntrs,quantity="FDR",arg=0.01,MLE=MLE)
```

returns a 95% confidence interval for the FDR at 0.01 based on the 95% confidence contours contained in the object *cntrs*. The returned interval is the range of *bum.FDR* (with 0.01 provided as the threshold of comparison) along the 95% contour of a and λ contained in *cntrs*. More information on *bum.CI* can be found by searching the code documentation for "Function: bum.CI".

References

Benjamini, Y. and Hochberg, Y. (1995), "Controlling the False Discovery Rate: a Practical and Powerful Approach to Multiple Testing." *Journal of the Royal Statistical Society B.* 57, 289-300.

Efron, B. et al. (2001) "Empirical Bayes Analysis of a Microarray Experiment." *The Journal of the American Statistical Association*, 96, 1151-1160.

Pounds, S and Morris, S. (2003), "Estimating the Occurrence of False Positives and False Negatives in Microarray Studies by Approximating and Partitioning the Empirical Distribution of p-values." Submitted to *Bioinformatics.*