Using the *confSAM* R package

Jesse Hemerik and Jelle Goeman

18 January 2016

Contents

Citing *confSAM*

If you use the *confSAM* package, please cite the paper *J. Hemerik and J. J. Goeman, False discovery proportion estimation by permutations: confidence for SAM* (submitted).

Introduction

The package *confSAM* is used for multiple hypothesis testing. It provides confidence bounds for the false discovery proportion in the context of SAM (Tusher, Tibshirani, and Chu 2001).

The false discovery proportion

Suppose hypotheses $H_1, ..., H_m$ are tested by calculating corresponding p-values $p_1, .., p_m$ and rejecting the hypotheses with small p-values. The number of false positive findings is then the number of hypotheses that are rejected even though they are true. The False Discovery Proportion (FDP) is this number divided by the total number of rejected hypotheses.

Instead of calculating a p-value for each hypothesis, it is also possible to calculate other test statistics, $T_1, ..., T_m$, say. One could then reject all hypotheses with test statistics e.g. exceeding some constant. (Possibly with a different constant for each hypothesis.)

In multiple testing it is often of interest to estimate how many of the rejected hypotheses are false findings. This is equivalent to estimating the FDP. The package *confSAM* allows estimation of this quantity.

As is usually the case with estimating quantities, providing a point estimate is not enough. What is also important is providing a confidence interval, so that one has e.g. 95% confidence that the quantity of interest lies in the interval. The package *confSAM* allows not only estimating the FDP, but also providing a confidence interval for it. More precisely, the package provides an confidence upper bound for the FDP, so that the user has e.g. 95% confidence that the FDP is between zero and this bound.

The package *confSAM* incorporates different methods for providing estimates and upper bounds. The methods vary in complexity and computational intensity. In the following it is explained how these methods can be used with the function `confSAM`.

Use of permutations

The methods in this package can be used if the joint distribution of the part of the data corresponding to the true hypotheses, is invariant under a group of permutations. For example, suppose that each test statistic T_i, $1 \leq i \leq m$, depends on some n-dimensional vector of obervations. Suppose for example that such a vector contains n gene expression level measurements: $n/2$ from cases and $n/2$ from controls. If the joint distribution of the gene expression levels corresponding to the true hypotheses is the same for cases and controls, then permuting the cases and controls does not change the joint distribution of the part of the data under the null. In that case the methods in this package can be used. For the precise formulation of this assumption, see Hemerik and Goeman 2017, Assumption 1 (submitted).

Designs with more than two groups or other transformations than permutations are also possible. See Hemerik and Goeman 2017 for general theory.

Basic estimate and bound

For the function `confSAM`, essentially the only input required is a matrix of p-values (or other test statistics). Every row of the matrix should correspond to a (random) permutation of the data.

Obtaining the matrix of test statistics

The *samr* package contains a function `samr` that allows computation of the test statistics as defined in their paper (Tusher, Tibshirani, and Chu 2001). More precisely, the object `tt` that `samr` returns, contains test statistics for the original data. Further, the object `ttstar0` contains a matrix of (unsorted) test statistics for the permuted version of the data. These objects can be used as input for our function `confSAM` (`ttstar0` should first be transposed).

Here we will not use *samr* to compute test statistics, but compute test statistics ourselves. As example data to work with, we consider the *nki70* dataset from the *penalized* package.

```
library(penalized)
data(nki70)
```

This survival data set concerns 144 lymph node positive breast cancer patients. For each patient there is a time variable and an event indicator variable (metastasis-free survival), as well as 70 gene expression level measurements. Using *confSAM* we will test the hypotheses $H_1, ..., H_{70}$ where H_i is the hypothesis that the expression level of gene i is not associated with the survival curve.

To be able to use `confSAM`, we now construct the required matrix of p-values. We will use random permutations, i.e. random reshufflings of the 144 vectors of gene expression levels. Hence we first set the seed.

```
library(survival)
set.seed(21983)
w<-100 # number of random permutations
pvalues <- matrix(nr=w,nc=70)
survobj <- Surv(time=nki70$time, event=nki70$event)
```

```
#compute the 70 p-values for each random permutation
for(j in 1:w){
  if(j==1){
    permdata <- nki70 #original data
  }
  else{
    permdata <- nki70[sample(nrow(nki70)),] #randomly shuffle the rows
  }
  for (i in 1:70) {
    form <- as.formula(paste("survobj ~ ", names(nki70)[i+7] ))
    coxobj <- coxph(form, data=permdata)
    sumcoxobj <- summary(coxobj)
    pvalues[j,i] <- sumcoxobj$coefficients[,5]
  }
}
```

```
## Warning in fitter(X, Y, strats, offset, init, control, weights = weights, :
## Loglik converged before variable 1 ; beta may be infinite.
```

We took the first permutation to be the original group labelling. Hence pvalues[1,] contains the p-values for the original data. Note that it is possible that the model assumptions are not exactly satisfied and the p-values corresponding to true hypotheses are not exactly uniform. A good property of the methods in *confSAM* however is that it does not matter if the p-values are exact for the methods to be valid (as long as they are computed in the same way for each permutation). The reason is that any test statistics are allowed.

Now that we have our p-value matrix, we are ready to use `confSAM`. Let us say that we will reject all hypotheses with p-values smaller than 0.03. Recall that `pvalues[1,]` contains the p-values for the original data. Thus the number of rejected hypotheses is found as follows:

```
sum(pvalues[1,]<0.03)
```

```
## [1] 17
```

The number of rejections is also automatically given by the function `confSAM`, as we will see below.

Basic confidence bound

We now turn to computing a basic confidence bound for the number of false positives (or equivalently, the FDP). By Hemerik and Goeman 2017 a basic $(1-\alpha)$-confidence bound for the number of false positives is the $(1-\alpha)$-quantile of the numbers of rejections for the permuted versions of the data. This basic upper bound is obtained as follows (for $\alpha = 0.1$):

```
library(confSAM)
confSAM(p=pvalues[1,], PM=pvalues, cutoff=0.03, alpha=0.1, method="simple")[3]
```

```
## Simple conf. bound for #fp:
##                            5
```

Here

- the argument p is the vector of p-values for the original unpermuted data;
- the argument PM is the matrix of p-values (or other test statistics);
- *cutoff* is the cut-off we have chosen. (*cutoff* is also allowed to be a vector of length `length(p)`, in which case the i-th p-value is compared to `cutoff[i]`.)
- *alpha* is such that $1-\alpha$ is the desired confidence level;

- *method="simple"* means that we want a simple upper bound (which is computationally fastest).

In our case, we made sure that the first row of the p-value matrix contained the original p-values. If we had not explicitly put the original p-values in the matrix (in which case they should be in the first row), then we should have included *includes.id=FALSE* in the function call.

Another argument that can be given to the function is *reject*, which can take the values *"small"*, *"large"* and *"absolute"*. This argument, together with the cutoff, determines which hypotheses are rejected. For example, the default is *"small"* and this means that all hypotheses with p-values (or test statistics) smaller than *cutoff* should be rejected. Setting *reject* to *"absolute"* means that all test statistics with absolute value greater than *cutoff* are rejected.

The above function call provides an upper bound for the number of false positives. Since we took $\alpha = 0.1$, we have 90% confidence that the number of false positives does not exceed this bound. To obtain a 90%-confidence upper bound for the false discovery proportion (FDP), we simply divide the above bound by the number of rejections.

Note that in the above we put '`[3]`' behind the function call. Leaving this out gives the full output of the function:

```
confSAM(p=pvalues[1,], PM=pvalues, cutoff=0.03, alpha=0.1, method="simple")
```

```
##               #rejections:     Simple estimate of #fp:
##                         17                           2
## Simple conf. bound for #fp:
##                          5
```

The first argument is the number of rejections, which we already computed above. The second argument is a basic median unbiased estimate of the number of false positives. (This is just the basic upper bound for $\alpha = 0.5$.) The function always produces these automatically since they require little additional computation time.

Improved upper bounds

Closed testing-based bound

Beside the basic upper bound discussed above, also more sophisticated upper bounds are derived in Hemerik and Goeman 2017. The first one we will discuss is related to the theory of closed testing. This method is the most computationally demanding, and is often infeasible when there are many hypotheses, but for the present example the method is feasible. The full closed testing-based method provides an upper bound that is often smaller than the basic upper bound discussed above, while still providing $(1-\alpha)100\%$ confidence. Details on how the bound is defined are in Hemerik and Goeman 2017.

To use this method we change the *method* argument into *full*.

```
confSAM(p=pvalues[1,], PM=pvalues, cutoff=0.03, alpha=0.1, method="full")
```

```
##                   #rejections:         Simple estimate of #fp:
##                             17                               2
## cl.testing-based bound for #fp:
##                              4
```

Note that the resulting upper bound is smaller than the bound obtained with *method="simple"*. In cases where the closed-testing method is infeasible, in some cases with many false hypotheses the basic bound can still be improved by setting *method="csc"*. In that case an exact, but conservative shortcut for the full closed testing-based method is used. A method that is more likely to improve the basic bound however, is the approximation method discussed below.

Approximation method

Setting *method="full"* can lead to computational infeasibility. In that case a possible solution is to approximate this bound. This is done by setting *method="approx"*. This corresponds to the approximation method detailed in Hemerik and Goeman 2017.

For example, in our example increasing the cut-off leads to more rejections, and it is a property of the method that it then tends to take longer or can be infeasible. To limit the computational burden, the approximation method can be used. We now increase the cut-off to 0.2 and illustrate the approximation method. For comparison we also compute the simple bound explained above.

```
#simple method
confSAM(p=pvalues[1,], PM=pvalues, cutoff=0.2, alpha=0.1, method="simple")
```

```
##                #rejections:     Simple estimate of #fp:
##                          40                          12
## Simple conf. bound for #fp:
##                          22
```

```
#approximation method
confSAM(p=pvalues[1,], PM=pvalues, cutoff=0.2, alpha=0.1, method="approx", ncombs=1000)[3]
```

```
## Appr. cl.testing-based bound for #fp:
##                                    17
```

The last result again means that with 90% confidence, the number of false positives does not exceed the stated bound. Note that the bound obtained with the approximation method is smaller than the simple bound.

Note above that an additional argument *ncombs* appears (default is 1000). This is the number of combinations that the approximation method checks per step. Details are in Hemerik and Goeman 2017. The higher *ncombs* is, the more reliable the bound is as a $(1-\alpha)$-confidence bound, but the longer the computation takes. We recommend taking *ncombs* very high ($> 10^4$) if time allows it.

References

Tusher, Virginia Goss, Robert Tibshirani, and Gilbert Chu. 2001. "Significance Analysis of Microarrays Applied to the Ionizing Radiation Response." *Proceedings of the National Academy of Sciences* 98 (9). National Acad Sciences: 5116–21.