Diagnostics for independent filtering

Wolfgang Huber

European Molecular Biology Laboratory (EMBL)

genefilter version 1.42.0 (Last revision 2013-03-30)

Contents

Abstract

This vignette illustrates the use of some functions in the *genefilter* package that provide useful diagnostics for independent filtering [1]. In particular, it provides diagnostics that are intended to help with

- the choice of filter criterion and
- the choice of filter cutoff.

1 Introduction

Multiple testing approaches, with thousands of tests, are often used in analyses of genome-scale data. For instance, in analyses of differential gene expression based on RNA-Seq or microarray data, a common approach is to apply a statistical test, one by one, to each of thousands of genes, with the aim of identifying those genes that have evidence for a statistical association of their expression measurements with the experimental covariate(s) of interest. Another instance is differential binding detection from ChIP-Seq data. The idea of *independent filtering* is to filter out those tests from the procedure that have no, or little chance of showing significant evidence, without even looking at their test statistic. Typically, this results in increased detection power at the same experiment-wide type I error, as measured in terms of the false discovery rate. A good choice for a filtering criterion is one that

1. is statistically independent from the test statistic under the null hypothesis,
2. is correlated with the test statistic under the alternative, and
3. does not notably change the dependence structure –if there is any– of the joint test statistics (including those corresponding to true nulls and to true alternatives).

The benefit from filtering relies on property 2, and we will explore it further in Section 3. Its statistical validity relies on properties 1 –which is simple to formally prove for many combinations of filter criteria with test statistics– and 3, which is less easy to theoretically derive from first principles, but rarely a problem in practice. We refer to [1] for further discussion on the mathematical and conceptual background.

2 Example data set

For illustration, we use the `pasillaGenes` dataset from the Bioconductor package *pasilla*; this is an RNA-Seq dataset from which we extract gene-level read counts for two replicate samples the were measured for each of two biological conditions: normally growing cells and cells treated with dsRNA against the *Pasilla* mRNA, which led to RNAi interference (RNAi) mediated knockdown of the Pasilla gene product.

```
> library("pasilla")
> data("pasillaGenes")
```

We perform a standard analysis with *DESeq* to look for genes that are differentially expressed between the normal and Pasilla-knockdown conditions, indicated by the factor variable `condition`. In the generalized linear model (GLM) analysis, we adjust for an additional experimental covariate `type`, which is however not of interest for the differential expression. For more details, please see the vignette of the *DESeq* package.

```
> library("DESeq")
```

```
> cds  = estimateSizeFactors( pasillaGenes )
> cds  = estimateDispersions( cds )
> fit1 = fitNbinomGLMs( cds, count ~ type + condition )
> fit0 = fitNbinomGLMs( cds, count ~ type  )
```

```
> res = data.frame(
+ filterstat = rowMeans(counts(cds)),
+ pvalue     = nbinomGLMTest( fit1, fit0 ),
+ row.names = featureNames(cds) )
```

The details of the anove analysis are not important for the purpose of this vignette, the essential output is contained in the columns of the dataframe `res`:

- `filterstat`: the filter statistic, here the average number of counts per gene across all samples, irrespective of sample annoation,
- `pvalue`: the test p-values,

Each row of the dataframe corresponds to one gene:

```
> dim(res)
```

```
[1] 14470     2
```

```
> head(res)
```

```
            filterstat pvalue
FBgn0000003      0.143 0.4601
FBgn0000008     54.714 0.9747
FBgn0000014      0.429 0.8395
FBgn0000015      0.857 0.7274
FBgn0000017   2507.429 0.0963
FBgn0000018    239.286 0.6794
```

3 Qualitative assessment of the filter statistic

First, consider Figure 1, which shows that among the approximately 40% of genes with lowest overall counts, `filterstat`, there are essentially none that achieved an (unadjusted) p-value less than 0.003 (this corresponds to about 2.5 on the $-\log_{10}$-scale).

```
> with(res,
+   plot(rank(filterstat)/length(filterstat), -log10(pvalue), pch=16, cex=0.45))
```

```
> trsf = function(n) log10(n+1)
> plot(ecdf(trsf(res$filterstat)), xlab=body(trsf), main="")
```

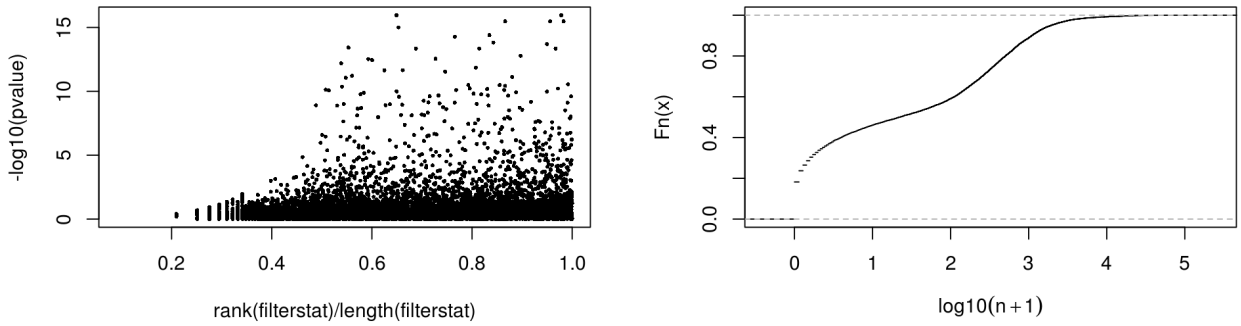

Figure 1: Left: scatterplot of the rank (scaled to $[0, 1]$) of the filter criterion `filterstat` (x-axis) versus the negative logarithm of the test `pvalue` (y-axis). Right: the empirical cumulative distribution function (ECDF) shows the relationships between the values of `filterstat` and its quantiles.

This means that by dropping the 40% genes with lowest `filterstat`, we do not loose anything substantial from our subsequent results.

For comparison, suppose you had chosen a less useful filter statistic, say, the gene identifiers interpreted as a decimal number. The analogous scatterplot to that of Figure 1 is shown in Figure 2.

```
> badfilter = as.numeric(gsub("[+]*FBgn", "", rownames(res)))
```

```
> plot(rank(badfilter)/length(badfilter), -log10(res$pvalue), pch=16, cex=0.45)
```

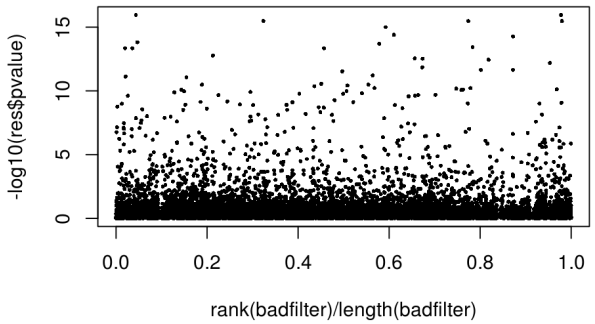

Figure 2: Scatterplot analogous to Figure 1, but with `badfilter`.

4 How to choose the filter statistic and the cutoff?

The `filtered_p` function in the *genefilter* package calculates adjusted p-values over a range of possible filtering thresholds. Here, we call this function on our results from above and compute adjusted p-values using the method of Benjamini and Hochberg (BH) for a range of different filter cutoffs.

```
> library("genefilter")
```

```
> theta = seq(from=0, to=0.5, by=0.1)
> pBH = filtered_p(filter=res$filterstat, test=res$pvalue, theta=theta, method="BH")
```

```
> head(pBH)
```

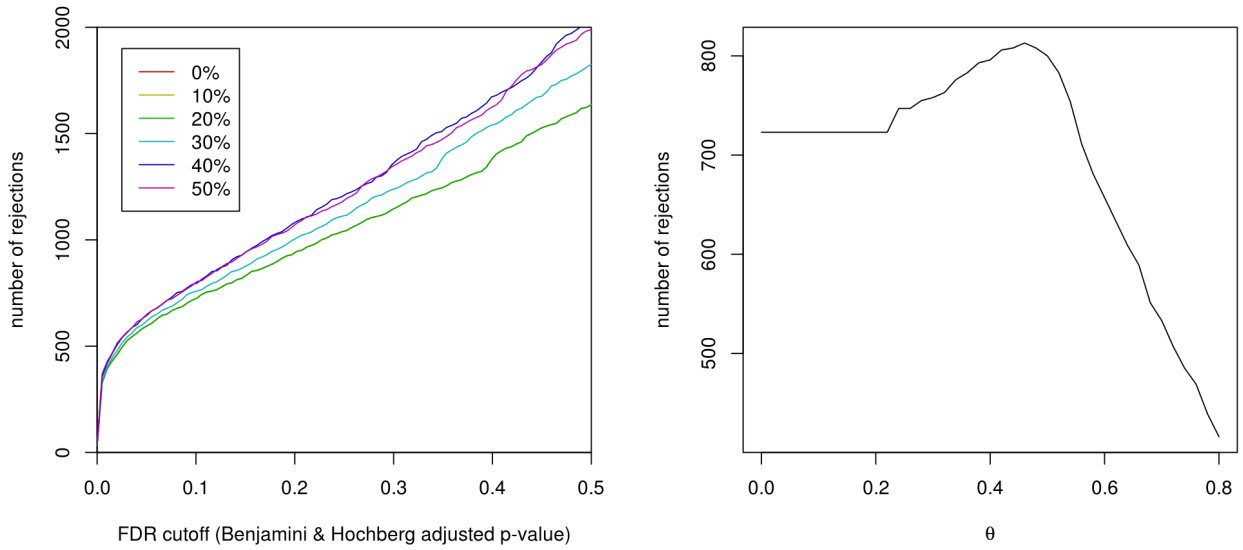

Figure 3: Left panel: the plot shows the number of rejections (i. e. genes detected as differentially expressed) as a function of the FDR threshold (x-axis) and the filtering cutoff θ (line colours, specified as quantiles of the distribution of the filter statistic). The plot is produced by the `rejection_plot` function. Note that the lines for $\theta = 0\%$ and 10% are overplotted by the line for $\theta = 20\%$, since for the data shown here, these quantiles correspond all to the same set of filtered genes (cf. Figure 1). Right panel: the number of rejections at FDR=10% as a function of θ.

```
        0%   10%   20%   30%   40%   50%
[1,] 0.895 0.895 0.895    NA    NA    NA
[2,] 0.997 0.997 0.997 0.998 0.995 0.993
[3,] 0.981 0.981 0.981    NA    NA    NA
[4,] 0.960 0.960 0.960 0.970    NA    NA
[5,] 0.593 0.593 0.593 0.517 0.452 0.412
[6,] 0.951 0.951 0.951 0.964 0.938 0.924
```

The rows of this matrix correspond to the genes (i. e., the rows of `res`) and the columns to the BH-adjusted p-values for the different possible choices of cutoff `theta`. A value of `NA` indicates that the gene was filtered out at the corresponding filter cutoff. The `rejection_plot` function takes such a matrix and shows how rejection count (R) relates to the choice of cutoff for the p-values. For these data, over a reasonable range of FDR cutoffs, increased filtering corresponds to increased rejections.

```
> rejection_plot(pBH, at="sample",
+                xlim=c(0, 0.5), ylim=c(0, 2000),
+                xlab="FDR cutoff (Benjamini & Hochberg adjusted p-value)", main="")
```

The plot is shown in the left panel of Figure 3.

4.1 Choice of filtering cutoff

If we select a fixed cutoff for the adjusted p-values, we can also look more closely at the relationship between the fraction of null hypotheses filtered and the total number of discoveries. The `filtered_R` function wraps `filtered_p` and just returns rejection counts. It requires you to choose a particular p-value cutoff, specified through the argument `alpha`.

```
> theta = seq(from=0, to=0.8, by=0.02)
> rejBH = filtered_R(alpha=0.1, filter=res$filterstat, test=res$pvalue, theta=theta, method="BH")
```

Because overfiltering (or use of a filter which is inappropriate for the application domain) discards both false and true null hypotheses, very large values of θ reduce power in this example:

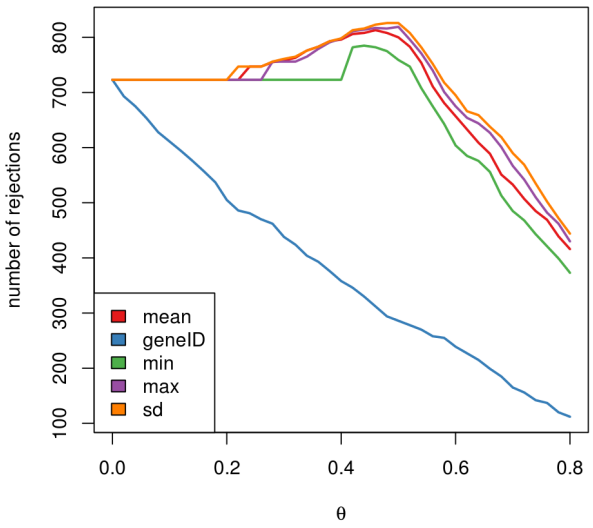

Figure 4: The number of rejections at FDR=10% as a function of θ (analogous to the right panel in Figure 3) for a number of different choices of the filter statistic.

```
> plot(theta, rejBH, type="l",
+      xlab=expression(theta), ylab="number of rejections")
```

The plot is shown in the right panel of Figure 3.

4.2 Choice of filtering statistic

We can use the analysis of the previous section 4.1 also to inform ourselves about different possible choices of filter statistic. We construct a dataframe with a number of different choices.

```
> filterChoices = data.frame(
+   `mean`   = res$filterstat,
+   `geneID` = badfilter,
+   `min`    = rowMin(counts(cds)),
+   `max`    = rowMax(counts(cds)),
+   `sd`     = rowSds(counts(cds))
+ )
> rejChoices = sapply(filterChoices, function(f)
+   filtered_R(alpha=0.1, filter=f, test=res$pvalue, theta=theta, method="BH"))
```

```
> library("RColorBrewer")
> myColours = brewer.pal(ncol(filterChoices), "Set1")
```

```
> matplot(theta, rejChoices, type="l", lty=1, col=myColours, lwd=2,
+         xlab=expression(theta), ylab="number of rejections")
> legend("bottomleft", legend=colnames(filterChoices), fill=myColours)
```

The result is shown in Figure 4. It indicates that for the data at hand, `mean`, `max` and `sd` provide similar performance, whereas the other choices are less effective.

5 Some more plots pertinent to multiple testing

5.1 Joint distribution of filter statistic and p-values

The left panel of Figure 1 shows the joint distribution of filter statistic and p-values. An alternative, perhaps simpler view is provided by the p-value histograms in Figure 5. It shows how the filtering ameliorates the multiple testing

problem – and thus the severity of a multiple testing adjustment – by removing a background set of hypotheses whose p-values are distributed more or less uniformly in $[0, 1]$.

```
> h1 = hist(res$pvalue[!pass], breaks=50, plot=FALSE)
> h2 = hist(res$pvalue[pass], breaks=50, plot=FALSE)
> colori <- c(`do not pass`="khaki", `pass`="powderblue")

> barplot(height = rbind(h1$counts, h2$counts), beside = FALSE,
+         col = colori, space = 0, main = "", ylab="frequency")
> text(x = c(0, length(h1$counts)), y = 0, label = paste(c(0,1)),
+      adj = c(0.5,1.7), xpd=NA)
> legend("topright", fill=rev(colori), legend=rev(names(colori)))
```

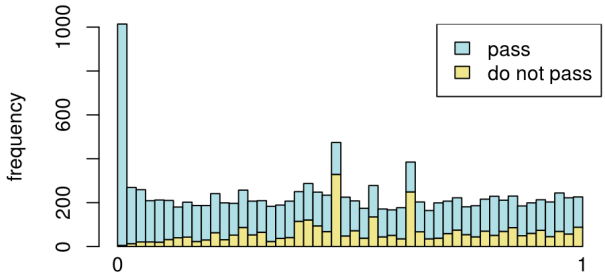

Figure 5: Histogram of p-values for all tests. The area shaded in blue indicates the subset of those that pass the filtering, the area in khaki those that do not pass.

5.2 Illustration of the Benjamini-Hochberg method

The Benjamini-Hochberg multiple testing adjustment procedure [2] has a simple graphical illustration, which we produce in the following code chunk. Its result is shown in the left panel of Figure 6.

```
> resFilt = res[pass,]
> orderInPlot = order(resFilt$pvalue)
> showInPlot = (resFilt$pvalue[orderInPlot] <= 0.06)
> alpha = 0.1

> plot(seq(along=which(showInPlot)), resFilt$pvalue[orderInPlot][showInPlot],
+      pch=".", xlab = expression(rank(p[i])), ylab=expression(p[i]))
> abline(a=0, b=alpha/length(resFilt$pvalue), col="red3", lwd=2)
```

5.3 Schweder and Spjøtvoll plot

Schweder and Spjøtvoll [3] suggested a diagnostic plot of the observed p-values which permits estimation of the fraction of true null hypotheses. For a series of hypothesis tests $H_1, \ldots, H_m$ with p-values p_i, they suggested plotting

$$(1 - p_i, N(p_i)) \text{ for } i \in 1, \ldots, m, \tag{1}$$

where $N(p)$ is the number of p-values greater than p. An application of this diagnostic plot to `resFilt$pvalue` is shown in the right panel of Figure 6. When all null hypotheses are true, the p-values are each uniformly distributed in $[0, 1]$, Consequently, the cumulative distribution function of $(p_1, \ldots, p_m)$ is expected to be close to the line $F(t) = t$. By symmetry, the same applies to $(1-p_1, \ldots, 1-p_m)$. When (without loss of generality) the first m_0 null hypotheses are true and the other $m - m_0$ are false, the cumulative distribution function of $(1-p_1, \ldots, 1-p_{m_0})$ is again expected to be close to the line $F_0(t) = t$. The cumulative distribution function of $(1 - p_{m_0+1}, \ldots, 1 - p_m)$, on the other hand, is expected to be close to a function $F_1(t)$ which stays below F_0 but shows a steep increase towards 1 as t

approaches 1. In practice, we do not know which of the null hypotheses are true, so we can only observe a mixture whose cumulative distribution function is expected to be close to

$$F(t) = \frac{m_0}{m} F_0(t) + \frac{m - m_0}{m} F_1(t). \tag{2}$$

Such a situation is shown in the right panel of Figure 6. If $F_1(t)/F_0(t)$ is small for small t, then the mixture fraction $\frac{m_0}{m}$ can be estimated by fitting a line to the left-hand portion of the plot, and then noting its height on the right. Such a fit is shown by the red line in the right panel of Figure 6.

```
> plot(1-resFilt$pvalue[orderInPlot],
+      (length(resFilt$pvalue)-1):0, pch=".", xaxs="i", yaxs="i",
+      xlab=expression(1-p[i]), ylab=expression(N(p[i])))
> abline(a=0, slope, col="red3", lwd=2)
> abline(h=slope)
> text(x=0, y=slope, labels=paste(round(slope)), adj=c(-0.1, 1.3))
```

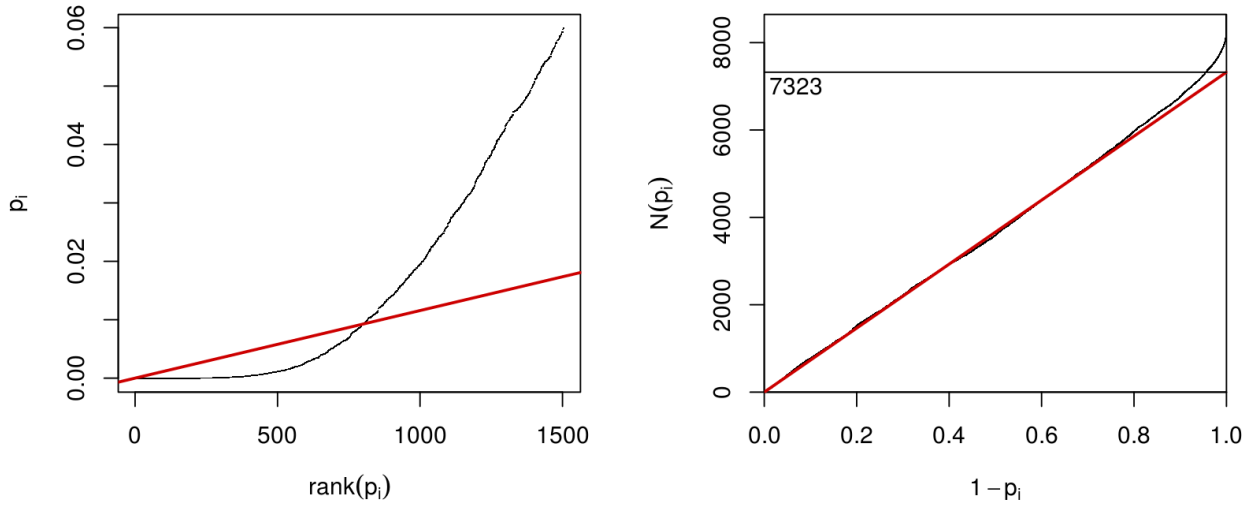

Figure 6: *Left:* illustration of the Benjamini-Hochberg multiple testing adjustment procedure [2]. The black line shows the p-values (y-axis) versus their rank (x-axis), starting with the smallest p-value from the left, then the second smallest, and so on. Only the first 1503 p-values are shown. The red line is a straight line with slope α/n, where $n = 8647$ is the number of tests, and $\alpha = 0.1$ is a target false discovery rate (FDR). FDR is controlled at the value α if the genes are selected that lie to the left of the rightmost intersection between the red and black lines: here, this results in 796 genes. *Right:* Schweder and Spjøtvoll plot, as described in the text.

Session information

- R version 3.0.0 (2013-04-03), `x86_64-unknown-linux-gnu`
- Base packages: base, datasets, grDevices, graphics, methods, parallel, stats, utils
- Other packages: Biobase 2.20.0, BiocGenerics 0.6.0, DESeq 1.12.0, DEXSeq 1.6.0, RColorBrewer 1.0-5, class 7.3-7, genefilter 1.42.0, lattice 0.20-15, locfit 1.5-9, pasilla 0.2.15
- Loaded via a namespace (and not attached): AnnotationDbi 1.22.0, Biostrings 2.28.0, DBI 0.2-5, GenomicRanges 1.12.0, IRanges 1.18.0, RCurl 1.95-4.1, RSQLite 0.11.2, Rsamtools 1.12.0, XML 3.96-1.1, annotate 1.38.0, biomaRt 2.16.0, bitops 1.0-5, geneplotter 1.38.0, grid 3.0.0, hwriter 1.3, splines 3.0.0, statmod 1.4.17, stats4 3.0.0, stringr 0.6.2, survival 2.37-4, tools 3.0.0, xtable 1.7-1, zlibbioc 1.6.0

References

[1] Richard Bourgon, Robert Gentleman, and Wolfgang Huber. Independent filtering increases detection power for high-throughput experiments. *PNAS*, 107(21):9546–9551, 2010.

[2] Y. Benjamini and Y. Hochberg. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society B*, 57:289–300, 1995.

[3] T. Schweder and E. Spjotvoll. Plots of P-values to evaluate many tests simultaneously. *Biometrika*, 69:493–502, 1982.