6 Permutation filtering

6.1 Introduction and outline

The computational step from scores to p-values as summarized in Equation (2.14) triggered our attention: to our knowledge, the transformation of scores to p-values in the special case of microarray data has not been studied in detail so far. In Section 2.4 we formally introduced the permutation approach to compute empirical p-values from observed scores. As a valid description of the null distribution is needed for the analysis of significance and this is difficult in case of microarray data, it is common practice to use simulated distributions obtained from randomizations of the original data (Dudoit *et al.*, 2002). Randomization is not only necessary because of the unknown distribution of intensity values but also due to special features of gene expression data: neither are genes independent from each other, nor do we know the underlying correlation structure. By randomly assigning class labels to patients and recomputing scores, we circumvent most ambiguities and generate a set of scores under the null hypothesis. The set of random scores serves as the null distribution, based on which we compute empirical p-values for the observed scores. Under the assumption that not a single gene is differentially expressed, one expects that this set of p-values is uniformly distributed.

A motivating example. To borrow information across genes, p-values are often computed using a pooled set of scores from all genes on the array (Storey and Tibshirani, 2003). The combined use of class label permutations and score pooling leads to a conceptual problem. In real applications, we will typically have both differentially and non-differentially expressed genes. While the permutations produce a justifiable null distribution for the non-differentially expressed genes, they produce a wider null distribution of scores for the differentially expressed genes. We show this with a simple simulation: for a set of 2500 genes and 14 patients

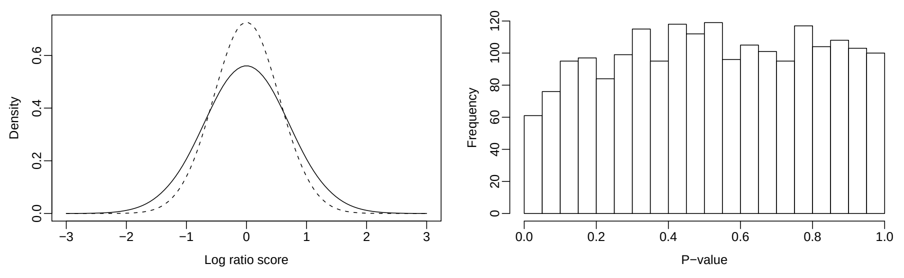

Figure 6.1: Effects of class label permutation and score pooling in simulated data set with 500 induced and 2000 non-induced genes. Left plot: Shown are averages of score densities of 500 non-induced (dashed line) and 500 induced genes (solid line). Although derived from all possible permutations of class labels, the densities of the induced genes have larger margins than the densities of the non-induced genes. Right plot: Histogram of p-values of the 2000 non-induced genes derived from score pooling. Low p-values are under-represented.

split into two equally sized groups, we generate random data drawn from a standard normal distribution. We slightly induce 500 genes by adding values drawn from a normal distribution with mean 1 for patients belonging to the first class. Next, we compute the log ratio score, that is the difference of class means, for each gene and each possible permutation of class labels. From the 14 patients, we yield $\frac{14!}{7!7!} = 3432$ possible permutations including the original one.

The left plot in Figure 6.1 shows averages of kernel density estimates of score distributions. The dashed line denotes the score density averaged over 500 non-induced genes while the solid line denotes the score density averaged over the 500 induced genes. Although both lines are based on all possible permutation scores and should therefore mirror the complete null distribution, the solid line has heavier tails than the dashed line. We conclude from this example that the permutation of class labels does not necessarily lead to an admissible null distribution for the differentially expressed genes.

When calculating p-values from pooled scores, the p-value distribution of non-induced genes is not uniform any more. We return to our simulation example above and compute p-values from the pooled set of all possible permutation scores of all genes. The right plot of Figure 6.1 shows the histogram of the resulting pooled p-values of the 2000 non-differentially expressed genes. Although we induced only one fifth of the gene set, the effect of the widened score distributions is visible: The p-value distribution of the non-induced genes is not uniform but lacks a certain amount of low p-values. The under-representation of small p-values is due to the fact that we include too many large (absolute) scores derived from induced genes into the calculation. Hence differentially expressed genes leave trace not only in the scores obtained from the original class labels, but also in pooled p-values obtained from comparisons with permuted class labels.

The problem discussed in literature. To our knowledge, the validity of single permutations has not been discussed in literature so far although the problem of choosing an appropriate null distribution has been studied extensively. There exist several approaches to improve permutation tests but none of them questions a single random permutation. Pollard and van der Laan (2003) evaluated combinations of re-sampling concepts and scores. The former divide into the common permutation approach and a bootstrap based re-sampling. The authors concluded that the choice of re-sampling method and score depends on the experimental design and further assumptions on the correlation structure. In a later simulation study, Pollard *et al.* (2005) introduced various bootstrap approaches to simulate a valid null distribution. Within each re-sampling concept however, bootstraps were randomly drawn and not restricted with regard to patients or genes.

Efron (2004) followed a different approach: given the set of observed scores, the author estimated the null density f_0 from the central peak of the mixture density f using smoothing splines and the normality assumption. Here genes in the center of the distribution have more influence on the estimated location and scale parameters than genes in the tails. Still, the approach depends on the normality assumption and on test settings with unimodal and symmetrical score distributions. Guo and Pan (2005) assigned weights to genes according to their estimated false discovery rate. Genes that appear to be significantly induced contribute less to the null score distribution than those genes that seem to be truly null. The authors suggested to iterate between the false discovery rate estimation and the assignment of weights. They showed that the weighted permutation scores improve the power to detect differentially expressed genes. Xie *et al.* (2005) examined the influence of differential and non-differential genes on the estimated null distribution using log ratio, t- or z-scores and suggested to exclude genes identified as differentially expressed from the estimation process.

In contrast to these approaches, we suggest to keep the whole set of genes but to not rely on an arbitrary set of random permutations. We motivate our proposal in the following section with a display of artifacts, which we commonly observe in microarray data. We derive a natural decision rule for valid permutations and introduce a simple filter algorithm in Section 6.3, which leads to a set of admissible permutations. The filtering effects the estimation of the global and local false discovery rate, and we explore the estimates' benefits in Section 6.4.

6.2 Artifacts in real data

Notation. We shortly review the notation of permutation methods from Section 2.4. Let matrix $\mathbf{X}$ be an $m \times n$ gene-expression matrix with genes in rows and patients in columns. Entry x_{ij} is the value of the ith gene observed for the jth patient with genes $i = 1, \dots, m$ and patients $j = 1, \dots, n$. In addition, we have a vector $\boldsymbol{c}_0 = (c_1, \dots, c_n)$ with c_j being the class label of the jth patient. Let $\boldsymbol{s}_0$ denote the vector of scores with entries $(s_{i0})_{i=1,\dots,m}$ and let $\boldsymbol{c}$ be a random permutation of the entries of vector $\boldsymbol{c}_0$. For each permutation of class labels $\boldsymbol{c}$, we recompute scores and derive a set of random scores $\boldsymbol{s}$. With B permutations and thus B permuted label sets $\boldsymbol{c}_1, \dots, \boldsymbol{c}_B$, this yields B random score vectors $\boldsymbol{s}_1, \dots, \boldsymbol{s}_B$. Let $\mathbf{S}$ be the $m \times (B+1)$ score matrix of the joint score vectors including the original one:

$$\mathbf{S} := (\boldsymbol{s}_0\, \boldsymbol{s}_1 \cdots \boldsymbol{s}_B) = (s_{ib}) \text{ with } i = 1, \dots, m \text{ and } b = 0, \dots, B. \tag{6.1}$$

The empirical p-value for score s_{i0} is then given as:

$$p_{i0} = \frac{1}{m(B+1)} \sum_{k=1}^{m} \sum_{l=0}^{B} I\{|s_{kl}| \geq |s_{i0}|\}. \tag{6.2}$$

Note that compared to Equation (2.13) the p-value got a second subscript "0". The second subscript refers to the class label vector from which it was derived. Here we computed the set of p-values $\boldsymbol{p}_0 = (p_{i0})_{i=1,\dots,m}$ belonging to the original classification $\boldsymbol{c}_0$. Otherwise the definition equals Equation (2.13). We summarize the computational steps from class labels $\boldsymbol{c}_0$ via scores $\boldsymbol{s}_0$ to p-values $\boldsymbol{p}_0$ into one function U_C defined as:

$$U_C(\boldsymbol{c}_0) = \boldsymbol{p}_0, \tag{6.3}$$

where $C = \{\boldsymbol{c}_0, \boldsymbol{c}_1, \dots, \boldsymbol{c}_B\}$ is the set of permutations.

Random permutations applied to data sets. We return to our six exemplary data sets and follow the same analysis scheme that lead to Figure 4.2: for each data set, we drew $B = 1000$ random permutations of the original labeling $\boldsymbol{c}_0$, computed the matrix **S** of z-scores and empirical p-values $\boldsymbol{p}_0 = U_{\mathcal{C}}(\boldsymbol{c}_0)$. Note that the data underlying Figures 4.2 and 5.3 were derived from this very set of 1000 random permutations.

Random permutations are assumed to guard against biological signals in the data. Hence we assume that the score matrix **S** derived from random permutations consists of random scores only. The more the observed scores deviate from the random scores, the more evidence for differential expression there is. This rationale forms the basis of all permutation approaches. However, it is only valid if we can surely rely on the set of random scores to be drawn from a valid null distribution. We apply a simple technique to uncover biological signals still contained in the random scores: we loop over the set of permutations $\mathcal{C} = \{\boldsymbol{c}_0, \boldsymbol{c}_1, \ldots, \boldsymbol{c}_B\}$ regarding each single permutation $\boldsymbol{c}_b$ in turn as the originally observed classification and compute p-values $\boldsymbol{p}_b = U_{\mathcal{C}}(\boldsymbol{c}_b)$ of the bth permutation. Hence, we use the function $U_{\mathcal{C}}$ not only for assigning a vector of p-values to the original class labels, but also to each permuted vector of class labels. Thus we map the score matrix **S** onto a p-value matrix **P**:

$$p_{ib} = \frac{1}{m(B+1)} \sum_{k=1}^{m} \sum_{l=0}^{B} I\{|s_{kl}| \geq |s_{ib}|\}, \tag{6.4}$$

for all genes $i = 1, \ldots, m$ and permutations $b = 0, \ldots, B$. For any random permutation $b \neq 0$ we require its p-value distribution to be uniform, hence giving no evidence for biological signal associated to $\boldsymbol{c}_b$. Our observations are different: in Panels A of Figures 6.2 to 6.7 we display the results of the mapping above applied to the $B = 1000$ permutations. The top left plot shows a multi-dimensional scaling (MDS) representation of the p-value distributions obtained by fixing single permutations. We derived the mapping into two dimensions from the Euclidean distances between the empirical cumulative distribution functions of the associated sets of p-values. Close points represent permutations $\boldsymbol{c}_b$, which produce similarly distributed p-values $\boldsymbol{p}_b = U_{\mathcal{C}}(\boldsymbol{c}_b)$.

In addition, we annotated up to four exemplary permutations by numbers including the original labels as no. 3, whose p-value distributions are shown in the top

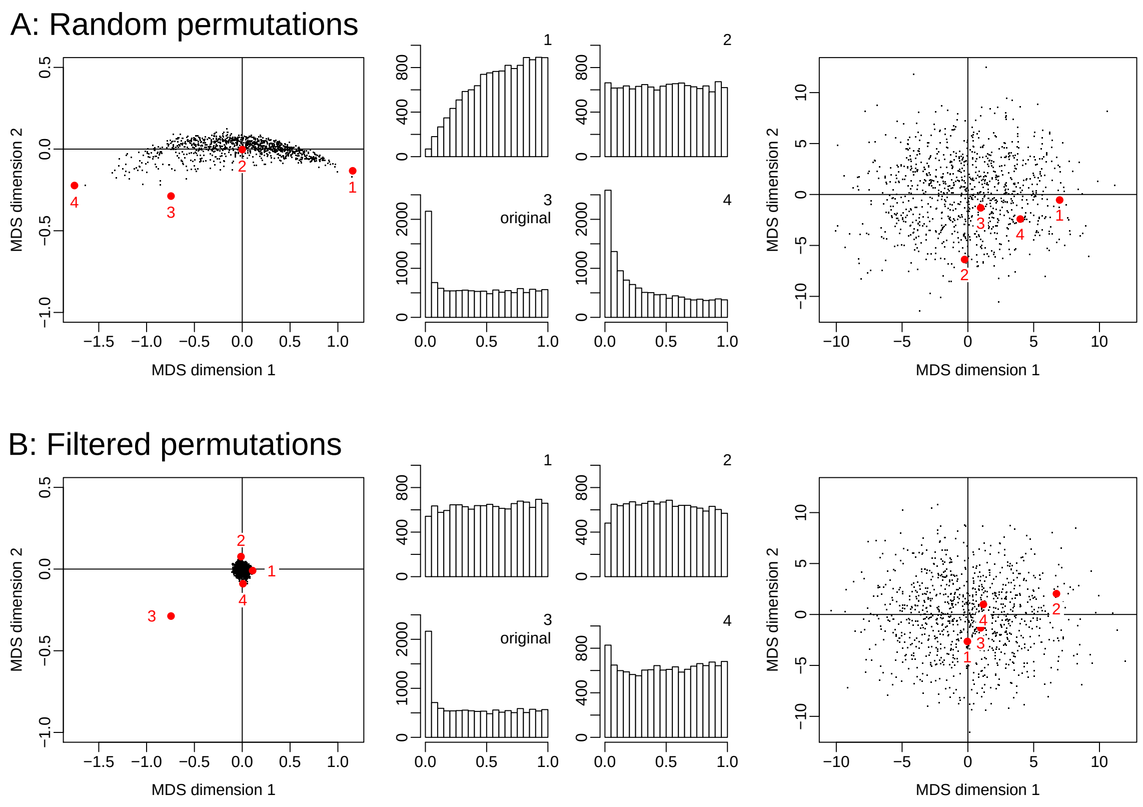

Figure 6.2: Effect of permutation filtering on ALL 1 data set. **A: Random permutation does not only produce valid null distributions.** The multi-dimensional scaling plot on the left-hand side shows distributional distances between 1000 sets of p-values resulting from random permutations. Euclidean distances between the cdf of the p-value sets were used. The four numbered examples show that permutations on the right side in the MDS plot have increasing densities, permutations on the left side have decreasing densities, and only permutations close to the origin produce uniform densities. No. 3 denotes the original class labels c_0. The scatterplot on the right-hand side shows a second MDS mapping of the permutations, now directly based on the Hamming distances of permuted class labels. The permutations do not cluster but scatter randomly around the origin. **B: Filtering of permutations leaves uniform p-value distributions.** The filtering algorithm returns 1000 permutations that produce uniform p-value distributions, which cluster around the origin in the MDS plot on the left-hand side. Again, no. 3 represents the original labeling c_0 while the other three permutations were chosen from the extremes of the filtered set to show that these are still admissible. The MDS plot based on Hamming distances between permutations is similar to the one in A. Filtered permutations still spread evenly in the permutation space.

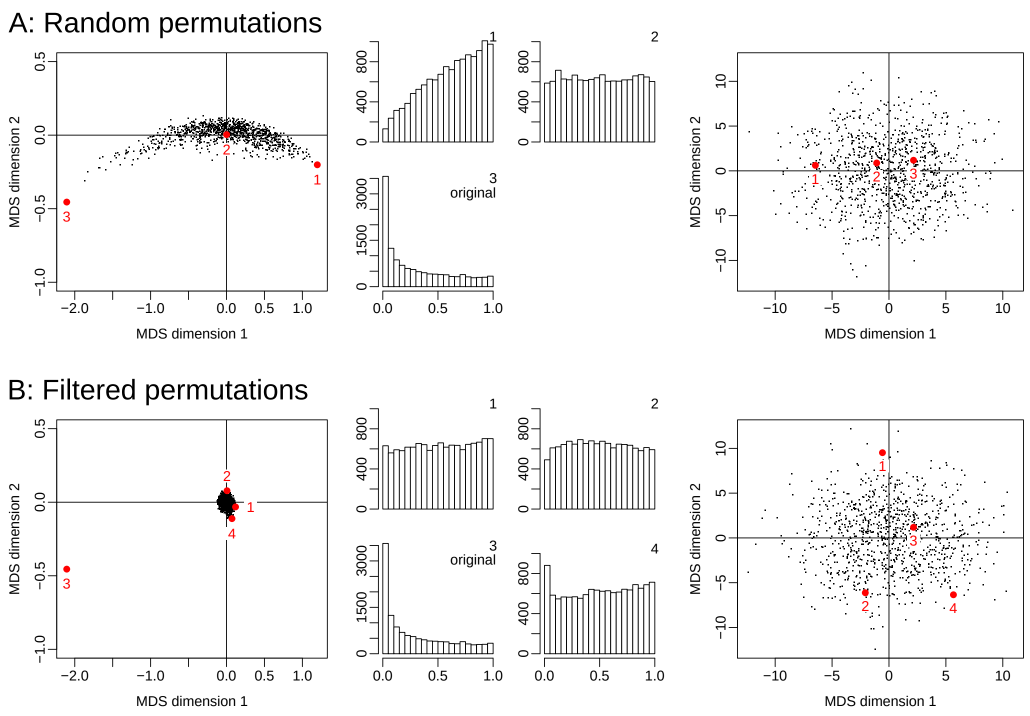

Figure 6.3: Effect of permutation filtering on ALL 2 data set. See Figure 6.2 for details.

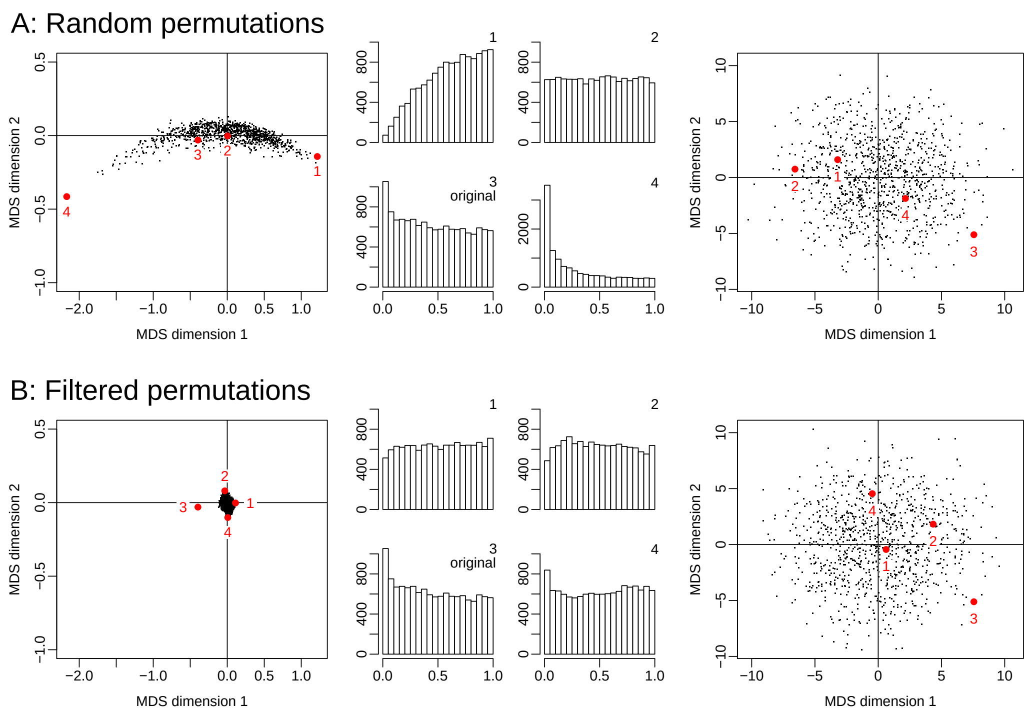

Figure 6.4: Effect of permutation filtering on Breast-cancer 1 data set. See Figure 6.2 for details.

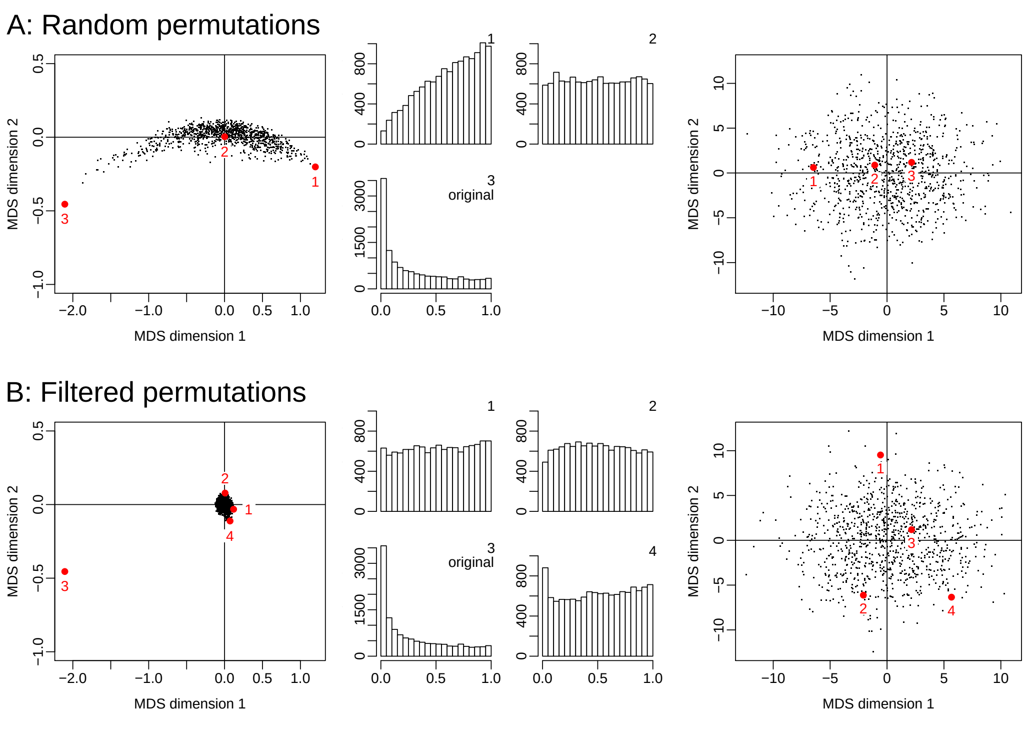

Figure 6.5: Effect of permutation filtering on Breast-cancer 2 data set. See Figure 6.2 for details.

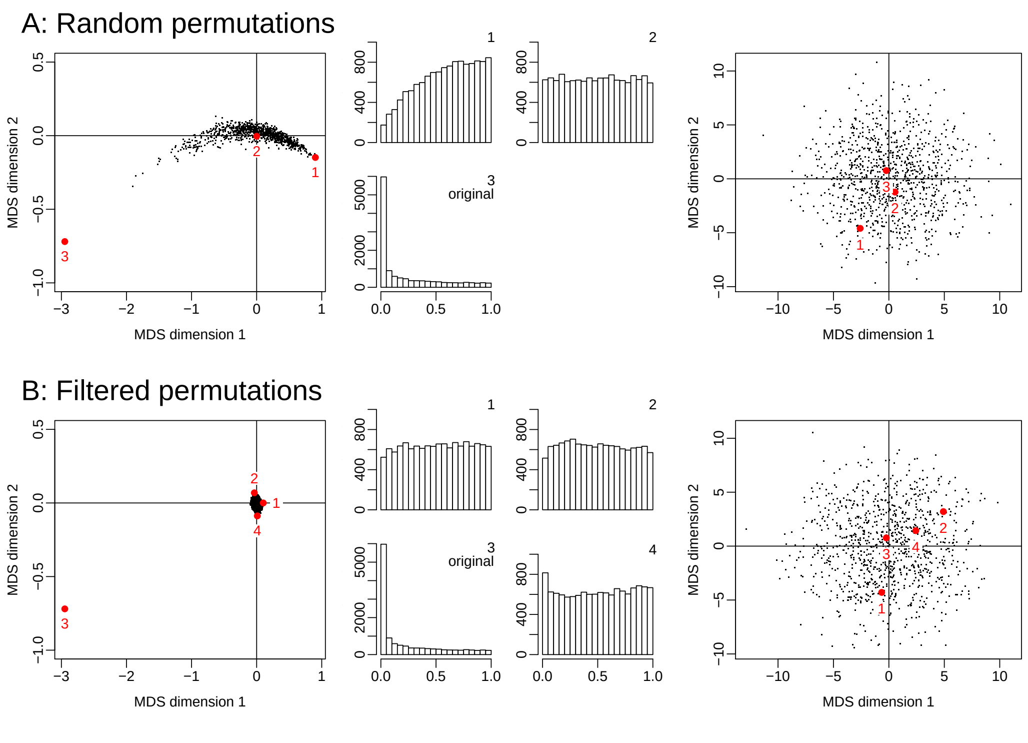

Figure 6.6: Effect of permutation filtering on Lung-cancer data set. See Figure 6.2 for details.

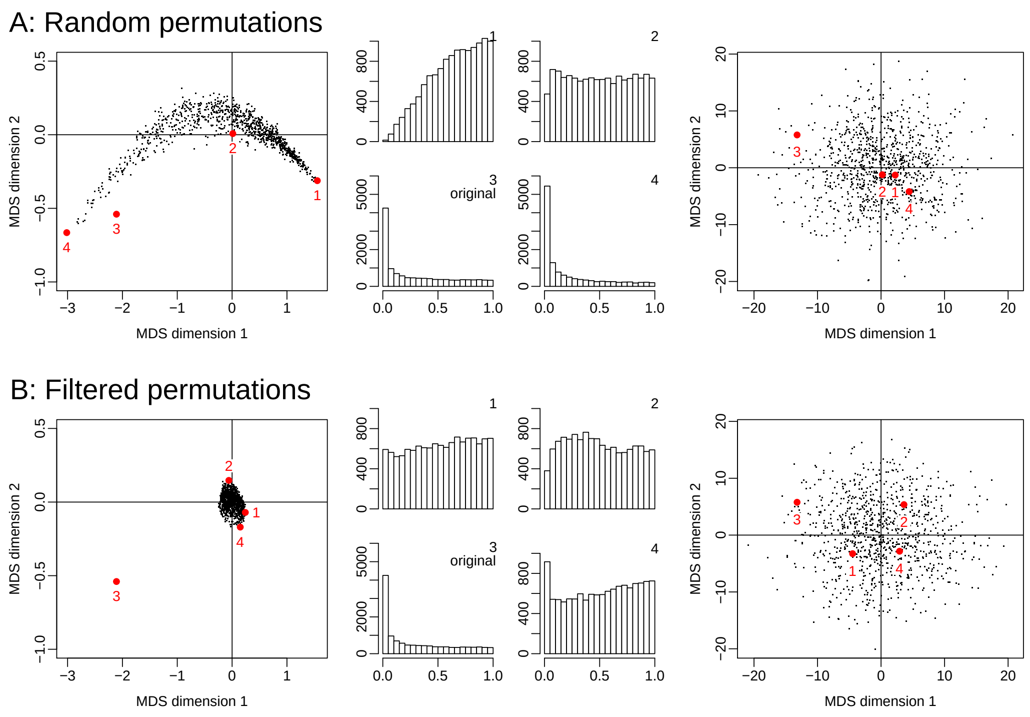

Figure 6.7: Effect of permutation filtering on Prostate-cancer data set. See Figure 6.2 for details.

middle plot. Only permutations close to the MDS origin produce uniform p-value distributions. The majority of permutations, however, deviates substantially from uniformity. In some experiments we find many permutations with p-value distributions deviating as much or even stronger from uniformity than the distribution of the original class labels, see example no. 4 in Panels A of Figures 6.2, 6.4 and 6.7. The banana-like shape of the MDS scatterplot can be observed in many data sets when random permutations are drawn. Usually one is left with a set of permutations with all kinds of increasing, uniform or decreasing p-value distributions, or—alternatively—with too narrow, null or too wide score distributions. In summary, a random draw of permutations does not guarantee that we base our final p-values p_0 on a set of scores that comply with a global null distribution.

6.3 A stochastic filtering approach

We propose a simple and well-performing search heuristic that returns a set of compliant permutations. The central idea is to transform not only the observed scores $\mathbf{s}_0$ to p-values but the whole score matrix $\mathbf{S}$. In particular, we apply function $U_{\mathcal{C}}$ to each vector $\boldsymbol{c} \in \mathcal{C}$ of permuted class labels, as was done in the previous section. We define $\mathcal{C}_0$ to be the set of *valid permutations*, which is a subset $\mathcal{C}_0 \subset \mathcal{C}$ of all permutations. For each permuted vector $\boldsymbol{c} \in \mathcal{C}_0$ we require the resulting p-values $\boldsymbol{p} = U_{\mathcal{C}_0}(\boldsymbol{c})$ to be uniformly distributed. This requirement must hold for all members of $\mathcal{C}_0$. If one permutation $\boldsymbol{c}^\star \in \mathcal{C}_0$ has a p-value distribution that deviates substantially from uniformity, either $\boldsymbol{c}^\star$ or large parts of the remaining permutations in $\mathcal{C}_0$ correlate with some non-random structure in the data. We propose the following filtering procedure to derive a set of permutations $\mathcal{C}_0$, which consistently produces uniform p-value distributions when calculating p-values for a fixed permutation using the remaining permutations in $\mathcal{C}_0$. The algorithm works iteratively to reduce computational time and to save memory. In each iteration, we draw a random set of permutations, compute scores and transform these to p-values. The p-value distribution of an individual permutation is tested for uniformity by applying the Kolmogoroff-Smirnoff score. We keep the permutations with best-fitting p-value distributions, that is with the smallest Kolmogoroff-Smirnoff scores. Note that these distributions are not necessarily *well*-fitting. However, if the number of possible permutations is large, there is a good chance of drawing permutations that follow the uniform distribution. The best permutations are kept in $\mathcal{C}_0$ and a new set of random permutations is drawn in the next iteration. Then, both sets are joined. Note that we do not have to recompute the scores of the kept permutations. We only have to recompute the p-values when we join $\mathcal{C}_0$ with the new set as the calculation is based on an altered set of permutations. In each iterative step, the number of permutations in $\mathcal{C}_0$ increases. Table 6.1 summarizes the algorithm.

Combinatorial remarks. As we keep increasing numbers of permutations in each round, the algorithm will always result in a set $\mathcal{C}_0$. As mentioned above, these permutations will in general comply with the global null hypothesis. If the patient numbers are small, it may happen that there do not exist enough unique and valid permutations such that $\mathcal{C}_0$ might contain non-compliant permutations. In the worst case, the number of wanted permutations exceeds the number of possible permutations. Then, our implemented version of the algorithm quits earlier and outputs the complete set of possible permutations accompanied by a warning message. Here the permutations are not filtered at all. However, they will be ordered according to their Kolmogoroff-Smirnoff score and one might either use a subset or reduce the number of wanted permutations. The filtering algorithm is implemented in our software package *twilight* from version 1.2.0 on.

Table 6.1: The filtering algorithm in detail.

1. Let $\mathcal{C} = \{\boldsymbol{c}_1, \ldots, \boldsymbol{c}_B\}$ be a set of unique random permutations of the original class labels $\boldsymbol{c}_0$. Apply function $U_{\mathcal{C}}$ to all $\boldsymbol{c}_b \in \mathcal{C}$, which yields the p-value vectors $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_B$. Choose a step size k and set $v = 1$.
2. Let F_b be the empirical cdf of the p-values in $\boldsymbol{p}_b$. Test each permutation for uniformity of its p-value cdf by computing the Kolmogoroff-Smirnoff statistic

   $$\mathrm{KS}_b = \max_{i=1,\ldots,m} |F_b(p_{ib}) - p_{ib}|.$$

   Keep the $v \cdot k$ permutations with the smallest KS statistic in the set $\mathcal{C}_0$. Increase $v = v + 1$.
3. Generate a new set of unique random permutations $\mathcal{C}$, join it with $\mathcal{C}_0$ and apply $U_{\mathcal{C}_0 \cup \mathcal{C}}$ to all $\boldsymbol{c}_b \in \mathcal{C}_0 \cup \mathcal{C}$.
4. Iterate steps 2 and 3 until $|\mathcal{C}_0|$ reaches a predefined number of permutations.
5. Compute the final vector of empirical p-values $\boldsymbol{p}_0 = U_{\mathcal{C}_0 \cup \boldsymbol{c}_0}(\boldsymbol{c}_0)$ for the original class labels.

6.4 Benefits of filtering

We apply the filtering algorithm as given in Table 6.1 to the exemplary data sets. In each case, we set the step size to $k = 50$, the number of new random permutations per iteration to 1000 and the stopping criterion to $|\mathcal{C}_0| \geq 1000$ unique permutations. With these default values, we need at least 20 iterations until the algorithm stops.

Permutation filtering produces valid null distributions. The effect of permutation filtering is shown in Panels B of Figures 6.2 to 6.7. The axes of the MDS plots equal those in Panels A. As expected, the filtered permutations lie closer to the origin. Again, we annotated the original labeling and three exemplary permutations. The latter were chosen from the margins of the cloud to show that even those produce acceptable uniform p-value distributions. We removed identical permutations within the iterative filtering. One might suspect that filtering introduces a selection bias in that the filtered permutations cluster strongly and do not spread over the entire permutation space. To show that this is not the case, we display a two-dimensional MDS mapping of the permutations. The mapping was derived from the Hamming distances between the binary vectors of permuted class labels before (Panel A) and after (Panel B) filtering. Filtered permutations do not form clusters but spread evenly over the permutation space in the MDS representation.

Permutation filtering leads to more significant genes. Permutations are the basis of any analysis of significance. From permutations, we compute p-values and from these q-values or estimates of the local false discovery rate. The filtering enhances the finding of significant outcomes. To show this, we compute p-values of the original labeling c_0 based on the random as well as on the filtered set of permutations. Note that these p-value sets correspond to the p-value histograms no. 3 in Figures 6.2 to 6.7 as well as in Figure 4.2. For all p-value sets, we estimated the positive false discovery rate as defined in Storey and Tibshirani (2003) but exchanged the prior $\widehat{\pi_0}$ with our SEP estimate. In the left-hand side plots of Figures 6.8 and 6.9, we display positive false discovery rate thresholds against the resulting size of gene lists. Filtering generally increases the number of significant genes in all cases. The increased number of significant genes is due to the removal of permutations with p-value distributions similar to that of the original labeling, that is with more small p-values than expected. These distributions correspond to score distributions with heavy tails. Their removal increases the empirical p-values of genes with high scores. The dashed lines in Figure 6.8 and 6.9 mark an exemplary threshold of $\mathrm{pFDR} \leq 0.05$. The sizes of the resulting gene lists are shown in the first column of Table 6.2. The last two columns again contain bootstrap estimates for the percentage of induced genes π_0 and 95% bootstrap confidence intervals as also shown in Table 5.5. Now, estimates are given for both random and filtered permutations.

Filtering intensifies the local false discovery rate on both sides of the twilight zone. We estimated the local false discovery rate based on the set of p-values from filtered and random permutations, and examined how the estimates differ with respect to filtering. We have already seen the non-filtered estimates in Figure 5.3. When plotting the curve of one minus local false discovery rate over the range of p-values we observe that filtering leads to a clearer representation of differential expression. Low p-values have higher posterior probabilities whereas the curve declines faster for increasing p-values than before. Possible twilight zones in the middle between clear differential and non-differential areas remain with decreased evidence. The right-hand side plots in Figures 6.8 and 6.9 show the posterior probability of differential expression over gene ranks. Filtering enhances the

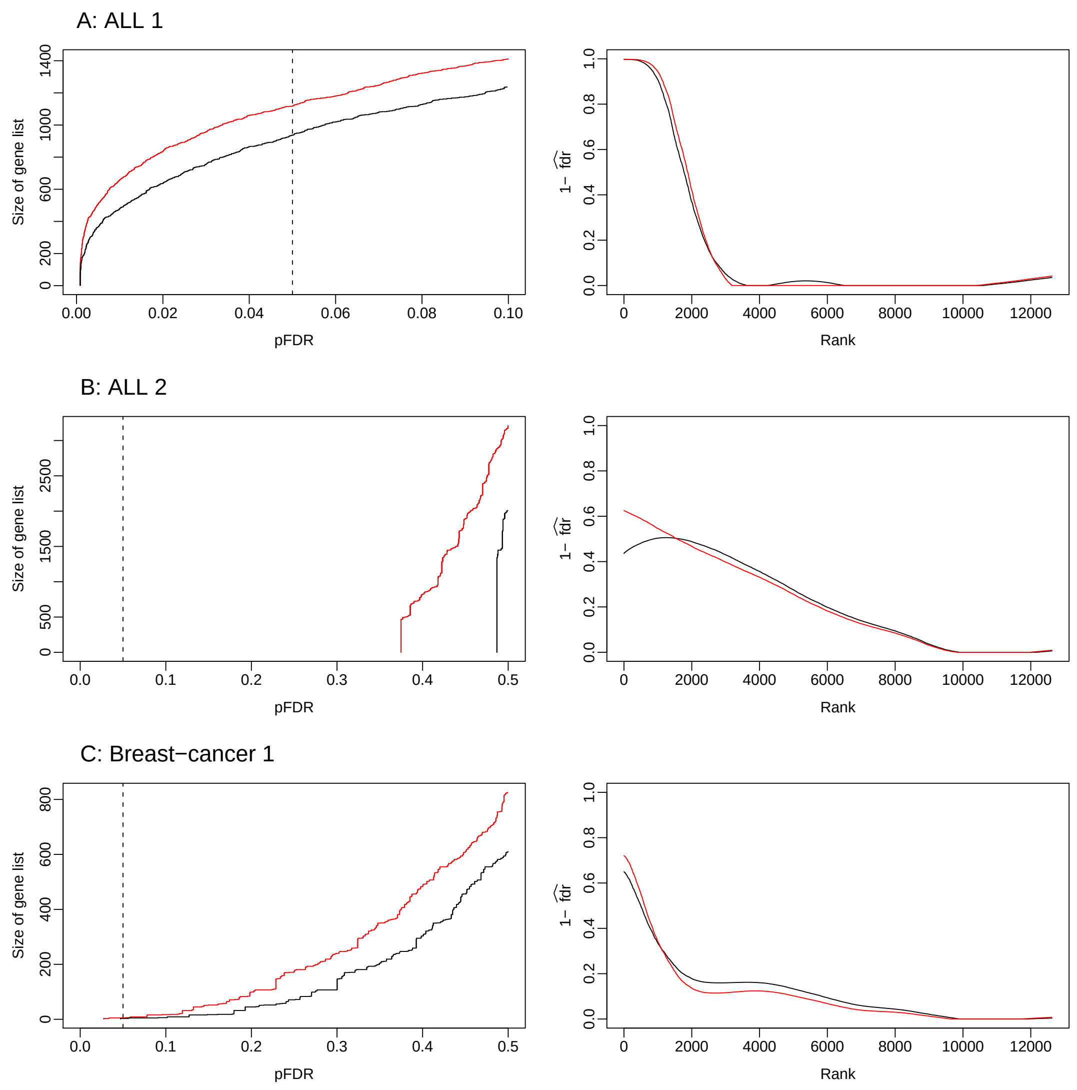

Figure 6.8: Permutation filtering leads to increased significance. On the left-hand side are positive false discovery rate thresholds plotted against the resulting size of gene list. The same pFDR threshold leads to more findings with filtering (red line) than without (black line). The vertical lines denote the arbitrary threshold of 5%, see Table 6.2. Right-hand side: The posterior probability of differential expression over gene ranks, again based on filtered (red line) or random permutations (black line). Filtering enhances the posterior probability for top genes, which levels off when the twilight zone is reached.

posterior probabilities of the top-most gene ranks but decreases it when the curve reaches the twilight zone.

Permutation filtering leads to a higher accuracy of the screening. So far, we examined the effect of filtering only on biological data. We observed an increase in significance and thus enhanced sensitivity but we do not know whether a significant gene is truly induced or not. If the increase in sensitivity brings along

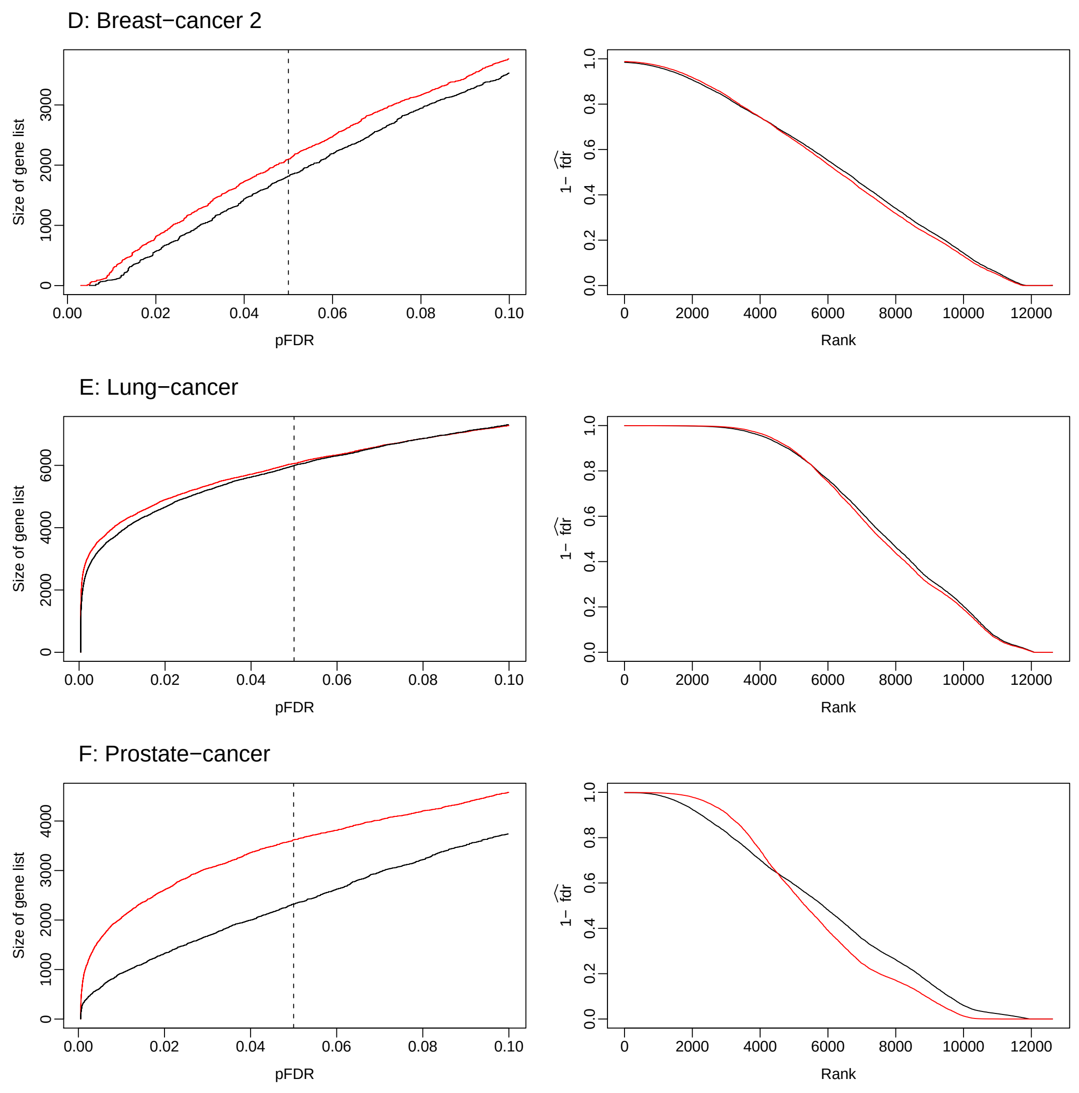

Figure 6.9: Permutation filtering leads to increased significance. See Figure 6.8 for details.

a highly decreased specificity and thus more false negatives, we do not gain anything. To show that this is not the case we used a simulation experiment where the true positives genes are known by design of the simulation. Of course, it is not possible to increase sensitivity and specificity simultaneously but we hope to keep a reasonable specificity while gaining sensitivity. To this end, we generated random data for 2500 genes and 10 patients per condition. First, we drew a vector of 2500 random values from a log normal distribution with location parameter 2 and scale parameter 0.3, and, taking these as mean values, generated 20 random samples from a normal distribution with variance 1. To induce the first 500 genes, we added a value of 2 to the samples of one condition. We introduced hid-

Table 6.2: Overview on exemplary data sets analyzed with or without permutation filtering. Shown are the numbers of genes with q-values below the desired levels, the bootstrap estimates for prior probability π_0, and the 95% bootstrap confidence intervals (CI) for π_0. The estimates are based on 1000 bootstrap samples within each comparison.

Comparison	Size of gene list with pFDR ≤ 0.05	Bootstrap $\widehat{\pi_0}$	Bootstrap 95% CI
ALL 1, random	937	0.8560	[0.8374, 0.8712]
ALL 1, filtered	1117	0.8574	[0.8411, 0.8699]
ALL 2, random	0	0.7803	[0.7476, 0.8045]
ALL 2, filtered	0	0.7832	[0.7546, 0.8072]
Breast-cancer 1, random	3	0.8818	[0.8538, 0.9080]
Breast-cancer 1, filtered	5	0.8962	[0.8694, 0.9200]
Breast-cancer 2, random	1816	0.4981	[0.4677, 0.5265]
Breast-cancer 2, filtered	2997	0.5045	[0.4779, 0.5300]
Lung-cancer, random	5992	0.3863	[0.3617, 0.4097]
Lung-cancer, filtered	6060	0.3937	[0.3678, 0.4189]
Prostate-cancer, random	2321	0.5360	[0.5046, 0.5613]
Prostate-cancer, filtered	3619	0.5562	[0.5307, 0.5772]

den non-random structure to the following 1000 genes by adding a value of 2 to five samples of each condition. Here the hidden variable simulated moderate induction. In a second setting, we added a value of 4 instead of 2 to simulate strong induction. Note that in either setting only the first 500 genes are differentially expressed between populations. We proceeded with the analysis as before and computed p-values based on 1000 filtered and 1000 unfiltered permutations. We ranked the genes by p-values and estimated the positive false discovery rate for every rank. We repeated the data generating procedure 100 times, each time calculating the number of truly induced genes within the list of genes with estimated pFDR ≤ 0.05.

Filtering increased the number of correctly identified genes for both moderate and strong hidden induction. Without filtering, the list of significant genes included an average of 390 true positive findings out of 500 truly induced genes in the strong setting. The filtering improved this number to 482 correctly identified genes on average. This difference was highly significant in a paired t-test ($p < 0.0001$). Speaking in relative values, the filtering increased the sensitivity, that is percentage of correctly identified induced genes, from 0.7809 to 0.9649. The specificity, that is percentage of correctly not-identified non-induced genes, decreased only slightly from 0.9998 to 0.9947. For the moderate setting, the val-

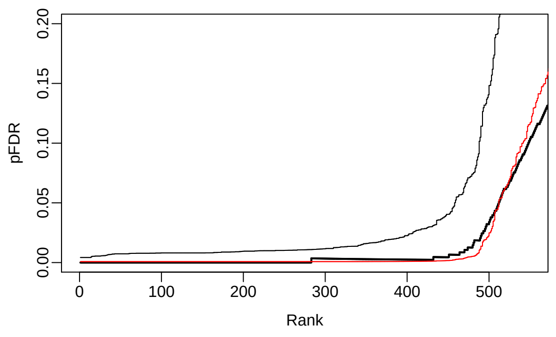

Figure 6.10: Permutation filtering leads to more accurate positive false discovery rate estimates. Ranks of high-scoring genes versus the true and estimated pFDR. The pFDR based on filtered permutations (thin red line) estimates the true pFDR (thick black line) with high accuracy for the first 500 ranks. The pFDR computed without filtering (thin black line) leads to conservative but inaccurate estimates.

ues were closer to each other but we still observed significant differences between the number of true positives, which were on average 459 without and 476 with filtering. The sensitivity improved from 0.9189 to 0.9529 while the specificity stayed virtually constant (0.9984 without and 0.9962 with filtering).

Permutation filtering produces more accurate estimates of the global false discovery rate. We used the simulation setting with strong induction to examine the accuracy of the estimates. In Figure 6.10 we display the ranks of the top-scoring genes with corresponding true and estimated positive false discovery rate. Again, the estimates are based on either a random or a filtered set of permutations. The thin black line depicts the non-filtered setting, which results in conservative estimates. The conservative behavior comes at the price of substantial overestimation of the true values. The estimates based on filtered permutations (red line) show less bias and match the true positive false discovery rate for the top-ranking genes well. For higher ranks beyond 500 they become conservative, too, which fits to our simulation setting of 500 truly induced genes.

In summary, we observe many benefits when basing the significance analysis on filtered permutations. The approach of excluding whole permutations instead of excluding or down-weighing genes or patients is novel. While keeping all genes and patients, it is an appealing way of building a valid null distribution for further analysis.