5 A novel estimator of the local false discovery rate

5.1 Outline

In Sections 3.4 and 3.5 we introduced the local false discovery rate and reviewed various estimation procedures. In the following Section 5.2 we introduce a novel estimator of the local false discovery rate, termed the *successive exclusion procedure*. We explicitly specify the estimation algorithm in Sections 5.2 to 5.4. In Section 5.5 we show the results of the local false discovery rate estimation for the exemplary data sets introduced in the previous chapter. We close this chapter with a comprehensive comparison of our approach to the estimators reviewed in Section 3.5.

5.2 A stochastic downhill search approach

The local false discovery rate is defined in terms of a mixture model that splits the observed p-value density f into a uniform part with rate π_0 and into a non-uniform part f_1 with rate $1 - \pi_0$. Recall from Section 3.4 that the mixture model is given as:

$$f(p) = \pi_0 + (1 - \pi_0) f_1(p). \tag{5.1}$$

In terms of the mixture model above, the local false discovery rate is then defined as:

$$\mathrm{fdr}(p) = \frac{\pi_0}{f(p)}, \tag{5.2}$$

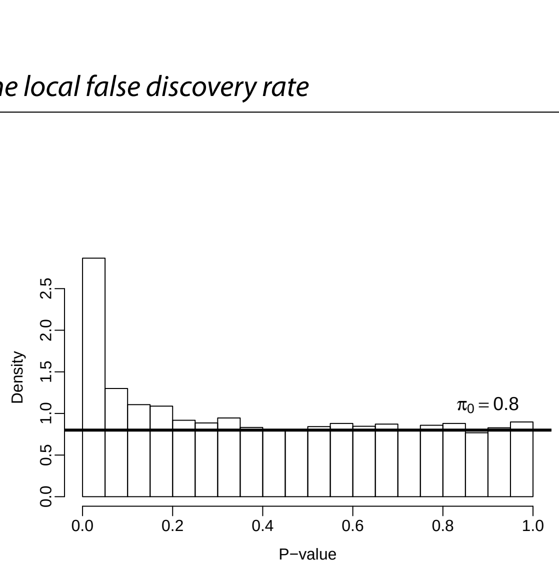

Figure 5.1: A p-value histogram derived from simulated scores with mixture parameter $\pi_0 = 0.8$. The amount of p-values belonging to non-induced genes is represented by the uniform block beneath the horizontal line. Given an estimate of π_0 and a density estimate of the mixture, it is easy to compute the local false discovery rate.

which is interpreted as the probability that a gene is not differentially expressed when observing its p-value p and conditioned on all observed p-values in the mixture f. In the following, we introduce a non-parametric estimator of the local false discovery rate based on appealingly simple foundations: to estimate the local false discovery rate, we have to know the percentage of non-induced genes π_0. Using the uniformity assumption for p-value distributions, prior π_0 fully specifies a uniform distribution and we are left with a density estimation of the mixture f. Plugging both estimates into Equation (5.2) yields the estimated local false discovery rate.

To estimate π_0, we split the set of observed p-values by dividing the p-value distribution horizontally into two parts, see Figure 5.1 for illustration. The two sets above and below the line represent p-values from induced and non-induced genes, respectively. Starting from the full set of p-values, we successively exclude some p-values until the remaining set looks like a typical draw from a uniform distribution. Of course, we cannot conclude that the individual genes corresponding to the p-values in the first set are *the* induced genes and those in the second set are *the* non-induced genes. However, the *size* of the uniform subset is a natural estimator for π_0. By computing a smoothed estimate of the mixture density, we yield the final estimates $\widehat{\mathrm{fdr}}(p)$ for all p-values p. The identification problem for π_0 is addressed by searching the *largest* set of genes such that the distribution of p-values can still pass as a sample from a uniform distribution. We call the algorithm "SEP" for *successive exclusion procedure.*

Formal description of the algorithm. We divide the set of observed p-values $(p_i)_{i=1,\dots,m}$ into two subsets. Let J denote the set of indices representing p-values that are assigned to the uniform subset. Let F_J be the empirical cumulative distribution function of the set of p-values $(p_i)_{i\in J}$. Our goal is to find the largest set J such that F_J is sufficiently close to a uniform distribution. For a given set J we measure the goodness-of-fit of the empirical to the uniform distribution using the Kolmogoroff-Smirnoff score defined as:

$$S(J) = \max_{i\in J}|F_J(p_i) - p_i|. \tag{5.3}$$

In addition, we need to include a size-dependent component that guarantees a high Kolmogoroff-Smirnoff score without removing more p-values than necessary from the uniform part. The result is a regularized fitting approach using an objective function composed of fit component $S(J)$ and a size component R_λ:

$$g(J,\lambda) = S(J) + R_\lambda(|J|) \tag{5.4}$$

$$\text{with} \quad R_\lambda(|J|) = \lambda\,\frac{m-|J|}{m}\log(m-|J|), \quad |J| < m, \tag{5.5}$$

where $R_\lambda(|J|)$ is strictly monotone in the size of the set J. Other choices of the penalty term R_λ are possible. The one above has proven to work well in applications.

The parameter $\lambda \geq 0$ determines the penalty on the regularization term. For $\lambda = 0$, we have $R_\lambda(|J|) = 0$ and hence the objective function only depends on the fit of the empirical distribution of the set $(p_i)_{i\in J}$ to the uniform distribution. Assume that no gene on the chip is induced. Here $|J| \approx m$ and the empirical distribution F_J resembles the uniform distribution but the fit is $g(J,0) \neq 0$ due to the sample variance of F_J. One can still find p-values to exclude such that F_J gets closer to identity, hence building distributions that are even more "uniform" in terms of goodness-of-fit than a typical draw from a uniform. This overfitting effect leads to a systematic underestimation of π_0. Note that in the overfitting phase we will only marginally improve the fit to the uniform, while $|J|$ and hence the resulting estimate $\widehat{\pi_0}$ can still change significantly. When choosing $\lambda > 0$, improving the fit by exclusion comes at a price in the size component $R_\lambda(|J|)$. Hence, the estimation of π_0 can be tuned by λ. We propose a calibration algo-

Table 5.1: The successive exclusion procedure in detail: Stochastic downhill search with fixed regularization parameter λ.

Let J be the index set of p-values representing the uniform part in the total set of p-values $(p_i)_{i=1,\dots,m}$.

1. Start with the full set $J = \{1,\dots,m\}$. Randomly select an index $i \in \{1,\dots,m\}$ and set $J = J\backslash\{i\}$.
2. Let F_J be the empirical cumulative distribution function of $(p_i)_{i\in J}$. Calculate the objective function

$$g(J,\lambda) = \max_{i\in J}|F_J(p_i) - p_i| + \lambda \frac{m-|J|}{m}\log(m-|J|).$$

3. Randomly select an index $i \in \{1,\dots,m\}$. If $i \in J$, set $J' = J\backslash\{i\}$. Else, set $J' = J \cup \{i\}$. Compute $g(J',\lambda)$.
   If $g(J',\lambda) < g(J,\lambda)$, set $J = J'$. Else, keep J unchanged.
4. Iterate steps 2 and 3 until the objective function was not reduced in $2m$ iterations.
5. Output final configuration J.

rithm for λ in the following section. For the time being, we assume the parameter λ to be fixed.

For a fixed λ, we need to minimize $g(J,\lambda)$ over all subsets J of the observed p-values, which is not feasible by exhaustive search. For heuristic optimization, we use a stochastic downhill approach. Assume, we have a candidate set J containing a subset of p-values. From the total set of p-values, we randomly draw a single p-value p_i. If the corresponding index i is already contained in J, we exclude i from J, which defines a new subset $J' = J\backslash\{i\}$. If i is not in J, we include it again: $J' = J \cup \{i\}$. For both subset configurations J and J', we compute the objective functions $g(J,\lambda)$ and $g(J',\lambda)$ according to Equation (5.4). If $g(J',\lambda) < g(J,\lambda)$, we substitute J with J', otherwise J remains unchanged. Starting with the full set of p-values, this procedure is iterated until the number of unsuccessful trials for a new configuration exceeds twice the total number of p-values m. Given the final configuration J, we yield an estimator for π_0 from the percentage of p-values in the uniform subset:

$$\widehat{\pi_0} = \frac{|J|}{m}. \tag{5.6}$$

Table 5.2: The successive exclusion procedure in detail: Estimation of the local false discovery rate.

Let J be the optimal subset of p-value indices found by the algorithm defined in Table 5.1.

1. Estimate the proportion of non-induced genes as $\widehat{\pi_0} = |J| \cdot m^{-1}$.
2. Divide the interval $[0, 1]$ into 100 bins derived from the 1%-quantiles of $(p_i)_{i=1,\dots,m}$. Compute histogram estimators $(h(l))_{l=1,\dots,100}$ for the density of $(p_i)_{i=1,\dots,m}$. For all l, set $q(l) = h(l)^{-1}$.
3. Apply a smoothing spline with seven degrees of freedom and decreasing weights $1/c(l)$ to $q(l)_{l=1,\dots,100}$, where $c(l)$ denotes the center of bin l. Compute the smoothed spline output in each $(p_i)_{i=1,\dots,m}$ and multiply with $\widehat{\pi_0}$. Truncate at 0 or 1 if values exceed the interval $[0, 1]$.

We divide the interval $[0, 1]$ into 100 bins derived from the 1%-quantiles of the total set of p-values. For each bin, we yield histogram estimators $(h(l))_{l=1,\dots,100}$ for the mixture density f. For all l, we compute the inverse $q(l) = h(l)^{-1}$. A pre-estimate of the local false discovery rate is derived by interpolating $(q(l))_{l=1,\dots,100}$ using a smoothing spline with seven degrees of freedom and decreasing weights $1/c(l)$, where $c(l)$ denotes the center of bin l. In our experience, this choice of smoothing parameters satisfyingly corrects for increasing variance of the histogram estimates. Finally, we multiply with the estimated prior $\widehat{\pi_0}$. A concise description of the SEP algorithm is given in Tables 5.1 and 5.2.

5.3 Calibration of the regularization parameter

The choice of the regularization parameter λ in Equation (5.4) is a trade-off between the exclusion of the smallest possible subset of p-values and the restriction to a p-value distribution that fits well to a uniform. Less regularization leads to an overly well fitting subsample but presumably excludes p-values that belong to the uniform part. On the other hand, a large penalty prohibits from excluding more values than necessary but results in an under-fitting subsample that is far from uniform. Hence, our strategy is to adaptively choose λ at the transition of over- and under-fitting such that only significant improvements of the fit component are accepted and overfitting is avoided. To this end, we select candidate values for λ and draw bootstrap samples of the set of p-values. To reduce computational time, the bootstrap samples have a smaller sample size than the observed set. For

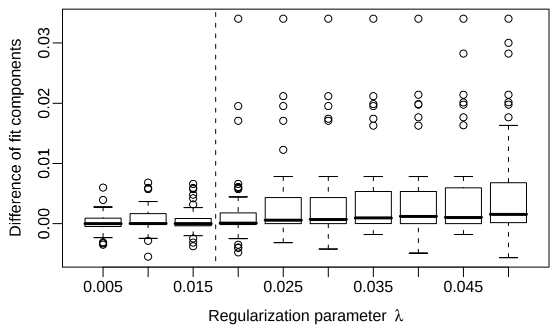

Figure 5.2: Effect of the regularization parameter λ. Shown are the differences of the fit components S computed with regularization to the fit computed without regularization. Each boxplot contains values from 50 bootstrap samples of size 1000 from a simulation with moderate induction ($\mu = 2$). The vertical line indicates the border between over- and under-fitting found by SEP. Here the largest admissible regularization parameter is $\lambda = 0.015$.

each candidate value, we run the SEP algorithm with the given penalty and report not the whole objective function but only the fit component S in Equation (5.4). Note that this is the fit computed *with* regularization. Hence, the higher the candidate value for λ is, the higher are the fit components indicating deviation from uniformity. Since we desire a good fit, we take the values of S found without regularization ($\lambda = 0$) as baseline. For each positive candidate value the baseline values are subtracted from the fit components of the bootstrap samples. Figure 5.2 shows exemplary boxplots of the resulting differences.

Now, we have to find an admissible value of λ such that the fit is not significantly worse than without regularization. To this end, we test whether the differences shown in Figure 5.2 deviate substantially from zero. Each set of bootstrap values is tested using the standard Wilcoxon ranksum test and the largest λ leading to non-significant differences is chosen as the final value. In the example above, the algorithm returns $\lambda = 0.015$. For larger values we observe that the differences generally increase such that the medians deviate from zero. The vertical line in Figure 5.2 denotes the transition between the over- and the under-fitting area. The boxplot belonging to $\lambda = 0.015$ is the last one on the left-hand side before the variance increases and outliers appear.

The calibration of λ is summarized in Table 5.3. Note that its description differs slightly from the first version published in Scheid and Spang (2004). We changed the algorithm since we observed problems when choosing λ from a set of p-values

Table 5.3: The successive exclusion procedure in detail: Calibration of the regularization parameter λ.

1. Draw 50 bootstrap samples $(p_i^B)_{i=1,\ldots,\min(1000,m)}$ from the given set of p-values. Set $\lambda_0 = 0, \lambda_1 = 0.005, \ldots, \lambda_{10} = 0.05$.
2. For λ_k, $k = 0, \ldots, 10$, run SEP as defined in Table 5.1 with $\lambda = \lambda_k$ on each bootstrap sample and keep the final configuration J^B. For each sample, output the fit component

$$S(J^B) = \max_{i \in J^B} |F_{J^B}(p_i^B) - p_i^B|.$$

3. For $k = 1, \ldots, 10$: Compute p-value p_k from a Wilcoxon ranksum test on the difference between fit values computed with $\lambda = \lambda_k$ and fit values computed with $\lambda = \lambda_0 = 0$.
4. Find first k such that $p_k \leq 0.05$ and set the regularization parameter to $\lambda = \lambda_{k-1}$.

where the uniform part is extremely small. In that case, many p-values have to be excluded to derive the uniform subset and hence λ has to be small enough to allow for this usually unwanted behavior. The updated version of the calibration algorithm performs well in simulations and also when applied to somewhat skewed clinical data sets. The problem of small uniform subsets also lead to the improvements described in the following section.

5.4 Fine-tuning

The regularization term is used to prevent from excluding more p-values than necessary from the uniform part. A strict penalization might lead to unwanted effects. We observed severe overestimation of π_0 when the true prior is close to zero. Here the mixture consists of a small percentage of uniformly distributed p-values and a huge over-representation of low p-values. We *must* exclude many genes, and those are probably more than the regularization term permits. Running SEP as defined in the previous sections leads to a conceptual al problem: if π_0 is too small, the algorithm stops too early because the penalty term increases quickly. Thus we exclude too few p-values such that the final estimate $\widehat{\pi_0}$ is misleadingly high. To prevent from SEP's misperformance on data sets with huge amount of differential expression, we propose the following two-step design.

Table 5.4: The successive exclusion procedure in detail: Fine-tuning of the π_0 estimate.

Let J be the index set of p-values representing the uniform part in the total set of p-values $(p_i)_{i=1,\dots,m}$.

1. Run the calibration as defined in Table 5.3 to yield the regularization parameter $\lambda = \lambda^\star$.
2. Start with the full set $J = \{1, \dots, m\}$.
3. Let F_J be the empirical cumulative distribution function of $(p_i)_{i \in J}$. Calculate the Kolmogoroff-Smirnoff score
$$S(J) = \max_{i \in J} |F_J(p_i) - p_i|.$$
4. If $S(J) \leq 0.25$, go to step 5.

   If $S(J) > 0.25$, run SEP as defined in Table 5.1 with $\lambda = 0$. Keep the final configuration as the initial set J and go to step 5.
5. Run SEP as defined in Table 5.1 with $\lambda = \lambda^\star$ and output the final configuration.

We start with finding a suitable regularization parameter as given Table 5.3. Then we evaluate how well the overall p-value distribution already fits to a uniform distribution. If it fits well, we run SEP with regularization. If the distribution is far from uniform, we split SEP into two separate rounds. First, we run it without regularization until the fit is well enough. Then we go on applying SEP with regularization. Hence, a small π_0 invokes two rounds of SEP, whereas we only perform one run when π_0 is assumed to be large or the p-value distribution is otherwise close to uniform. The fine-tuned algorithm as summarized in Table 5.4 has proven to work well in applications.

5.5 Features and applications

We proceed with discussing main features of the SEP algorithm and applying it to the six exemplary data sets introduced in Chapter 4. The general performance of our algorithm is evaluated in the following section where we compare the approach to its competitors introduced in Section 3.5.

Stability of solution. The core of SEP is the downhill search routine, which evaluates the benefit of inclusion or removal of a single p-value in each step. This regime leads us to a local minimum of the objective function where the price paid for the penalty term exceeds the improvement gained in the fit component. Running the algorithm with different starting values results in stable estimates. A single run of SEP might still fail simply by chance. We propose to apply SEP ten times with different starting points and to report averaged estimates. To assess the variability of the estimates, we run the algorithm on bootstrap samples of the p-value set and produce bootstrap averages as point estimators and bootstrap confidence intervals as measures of uncertainty.

Performance for non-induced genes only. To assess the performance under the complete null hypothesis, we run the algorithm on random p-values drawn from the uniform distribution representing experiments on non-induced genes only. Here we want to exclude as few p-values as possible to estimate the prior probability π_0 close to the target value 1. This setting evaluates the performance of the penalty term or rather the procedure that calibrates the parameter λ. We randomly draw 10000 values from a uniform distribution, apply SEP and repeat this procedure 100 times. The percentage π_0 is estimated to be 0.9848 on average with a standard deviation of 0.0136.

Application of SEP. We start with applying the calibration algorithm to derive optimal penalty parameters λ. We then run SEP on the complete sets of p-values with 1000 bootstrap samples each. In Table 5.5 we summarize some SEP results for the six data sets. Shown are the optimal values for λ and the estimated percentages π_0 of genes being non-induced. The estimates are similar to the estimated percentages derived previously with the method of Storey (2003), see Table 4.1. The 95% bootstrap confidence intervals are tight, deviating from the mean percentage $\widehat{\pi_0}$ at most $\pm 3\%$.

In Figure 5.3 we display the resulting curves of estimated posterior probabilities of differential expression with 95% bootstrap confidence intervals. As observed above, the amount of differential expression deviates substantially between comparisons. In case of the ALL 1 data set, the posterior probability of differential expression decreases rapidly. Genes corresponding to low p-values have a high probability of being induced. The situation changes for genes with low p-values in the ALL 2 and the Breast-cancer 1 data set. Here the highest estimated probabilities are around 0.5 for ALL 2 and 0.7 for Breast-cancer 1 indicating that even

Table 5.5: Application of the SEP algorithm. Shown are the optimal values for the regularization parameter λ, 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	Optimal λ	Bootstrap $\widehat{\pi_0}$	Bootstrap 95% CI
ALL 1	0.005	0.8560	[0.8374, 0.8712]
ALL 2	0.005	0.7803	[0.7476, 0.8045]
Breast-cancer 1	0.010	0.8818	[0.8538, 0.9080]
Breast-cancer 2	0.010	0.4981	[0.4677, 0.5265]
Lung-cancer	0.025	0.3863	[0.3617, 0.4097]
Prostate-cancer	0.015	0.5360	[0.5046, 0.5613]

apparently highly significant genes have to be explored with care. For ALL 2, we observe a slight increase in probability at the beginning. For Breast-cancer 1, a plateau around the p-value level of 0.2 is visible. The curves of the remaining comparisons Breast-cancer 2, Lung-cancer and Prostate-cancer moderately decrease from one to zero. In contrast to the ALL 1 data, these three comparisons exhibit broad twilight zones where the posterior probabilities decline slowly towards zero.

5.6 Simulation and results

The performance of our novel algorithm was evaluated in a comprehensive simulation. We compared SEP to the set of π_0 estimators given in Section 3.5. To reduce computation time, the final SEP estimate of π_0 is not an average of ten runs with different starting points but derived from a single run. In addition we applied SEP without regularization ($\lambda = 0$), which provides further information on the effect of the penalty term in Equation (5.5). We refer to SEP with regularization as Sep and to SEP without regularization as Sep0. For both settings, we used core functions of our implementation in package *twilight* introduced in the following section.

Simulation settings. The performance of the estimation procedures was evaluated with respect to three parameters: the number of genes m, the strength of induction μ, and the percentage of non-induced genes π_0. The number of genes was set to $m = 1000$ and $m = 10000$. The scores were drawn independently from

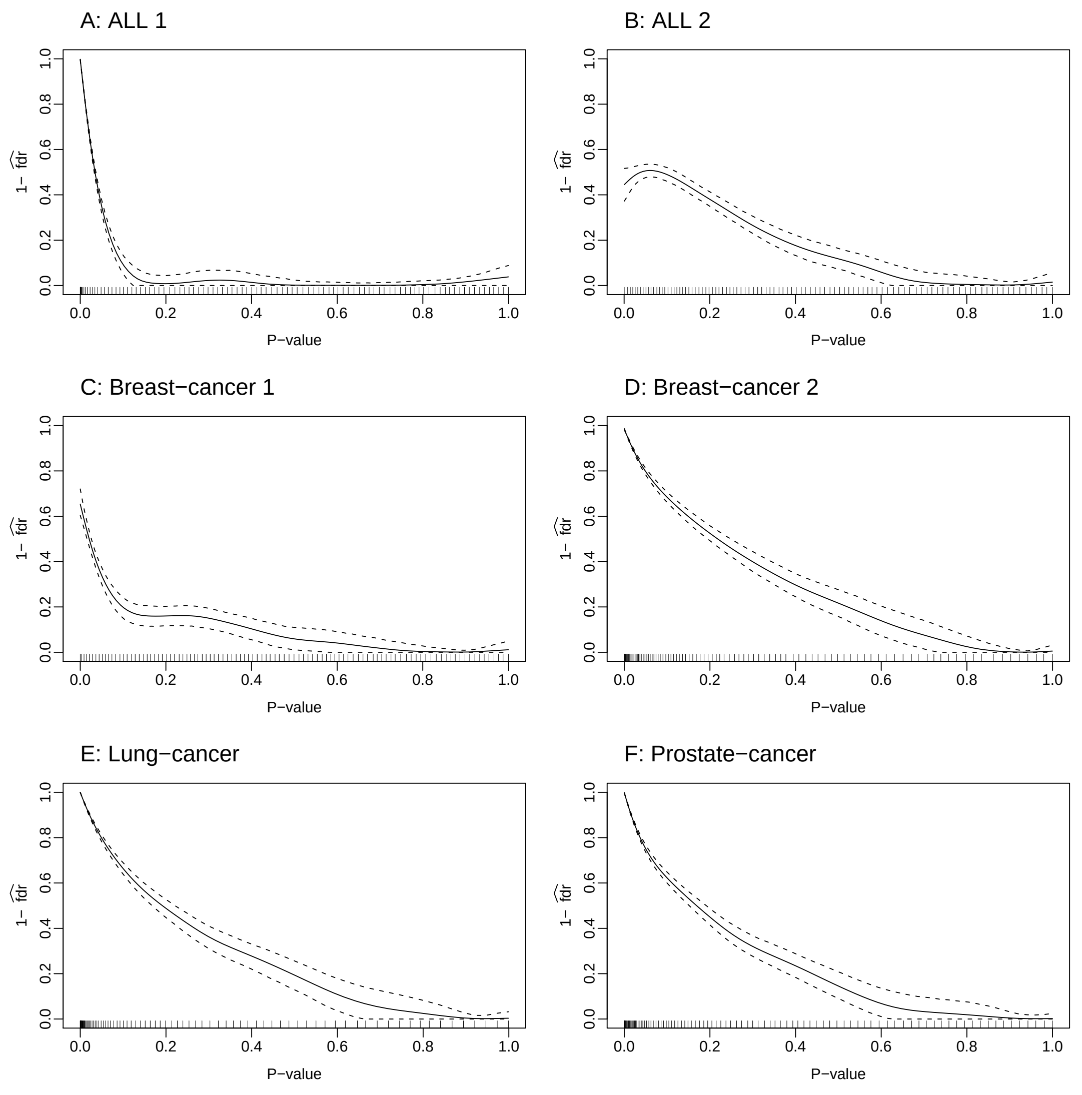

Figure 5.3: Estimates of the local false discovery rate for the exemplary data sets. Accuracy of the estimates was evaluated on 1000 bootstrap samples. The solid lines are bootstrap means, the dashed lines are 95% bootstrap confidence intervals. Bottom ticks represent 1%-quantiles of observed p-values.

a standard normal distribution $\mathcal{N}(0,1)$ at rate π_0, and from a normal distribution $\mathcal{N}(\mu,1)$ at rate $\pi_1 = 1 - \pi_0$. The location parameter μ models the strength of induction and varied from 0 to 4 by steps of 0.25. The percentage of non-induced genes π_0 varied from 0.5 to 0.95 by steps of 0.05. We included $\pi_0 = 0.99$ as well to evaluate the performance when the p-values resemble almost a uniform distribution.

For each combination (π_0, μ, m), we drew normal scores as described above and converted these to p-values under the null distribution $\mathcal{N}(0,1)$. The set of p-values was passed on to the estimators yielding either $\widehat{\pi_0}$ directly or $\widehat{\pi_1}$, which was then changed to $\widehat{\pi_0} = 1 - \widehat{\pi_1}$. The estimates were truncated if they exceeded the interval $[0, 1]$. The p-value conversion was not done for method Locfdr, where we passed on the normal scores. For each parameter combination and procedure, we did 100 repetitions with randomly drawn scores. We observed a substantial increase in computation time for method Genemix although the recommended number of iterations had been reduced. A complete run was not feasible for $m = 10000$ genes. Thus we shortened the simulation for Genemix to the parameter combinations shown in Figure 5.5.

The final output of each single run was the ratio $\widehat{\pi_0}/\pi_0$ of the estimated to the true percentage. An unbiased estimator should yield ratios close to one. Ratios above one correspond to overestimation of π_0, which is less critical than underestimation (ratios smaller than one). For evaluation, we first depict boxplots of the estimates with μ and m fixed and π_0 varying. A reasonable estimator produces ratios close to one with small variance and, if not exactly returning target 1, with a tendency of overestimation. In general one expects the estimators to perform better at higher levels of induction as the separation between induced and non-induced genes becomes clearer. Second, the ratios should converge towards one with increasing percentage π_0 since the estimation bias is of less importance. Third, the variance of the estimates should decrease with an increasing number of genes. For enhanced comparison, we truncated the plotting region of the following boxplots to the ratio interval $[0.75, 1.25]$.

The poor performers. We start the result section with removing those estimators from further consideration that are biased downwards. The poor performers are Pre, Bum, and Splosh yielding ratios smaller than one. Splosh starts well but underestimates π_0 with increasing true π_0. In Figure 5.4 we display the corresponding values over the range of π_0 with $\mu = 3$ fixed. With π_0 fixed, the three estimators show similar behavior: they are biased upwards for small μ but run into underestimation problems for large μ (data not shown).

The moderate performers. Some methods show artifacts or perform visibly worse than the good estimators. We decided to narrow down our set of good candidates by excluding the moderate performers as well. Figure 5.5 displays the ratios of methods Locfdr, Genemix, Lbe, and Storey. Method Locfdr returns upper bounds on π_0, which overestimate π_0 severely when the true π_0 is small. An estimated bound of $\widehat{\pi_0} = 1$ does not provide any information when we

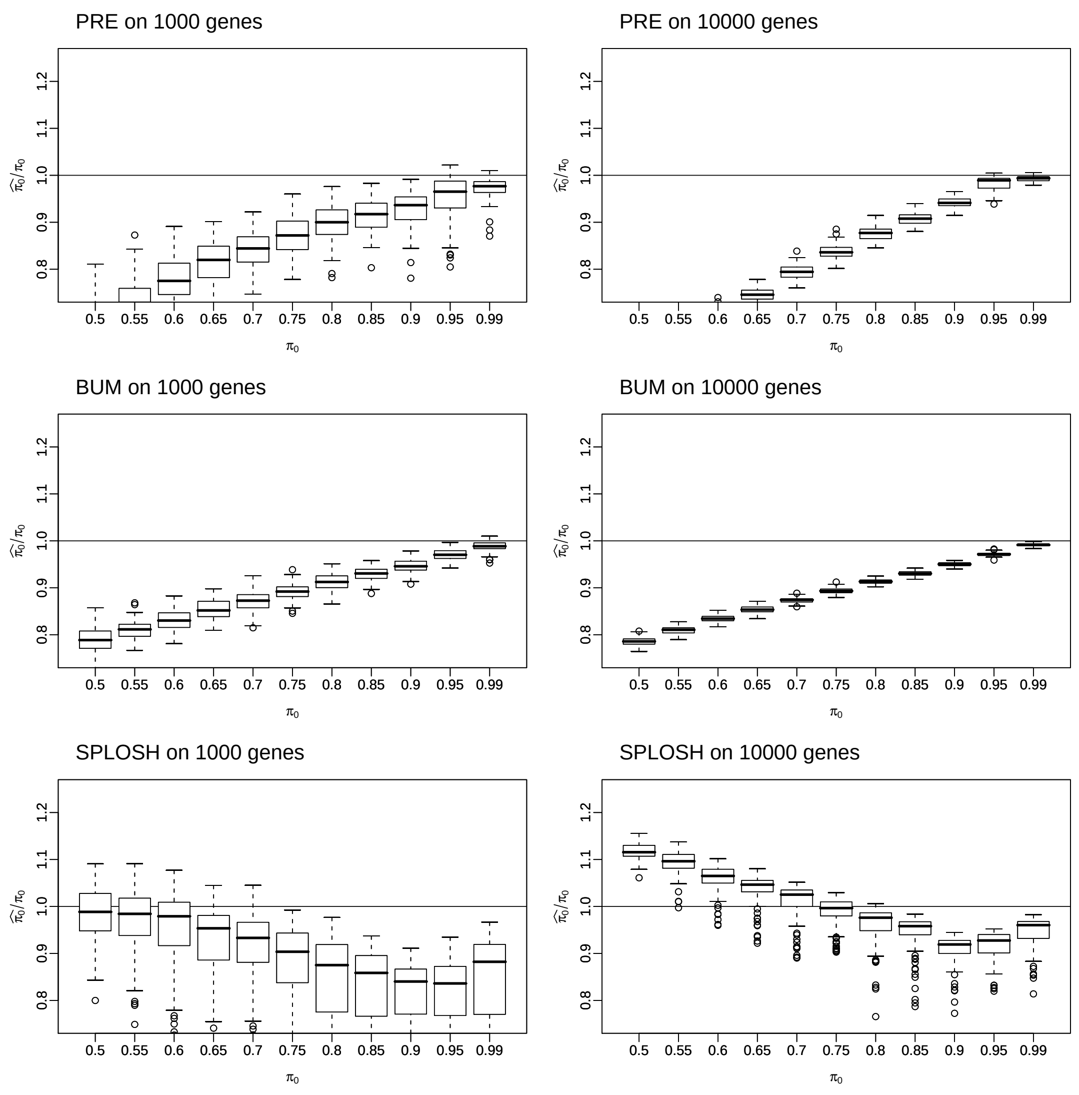

Figure 5.4: Distribution of ratio $\widehat{\pi_0}/\pi_0$ for $\mu = 3$ and varying π_0. Shown are the simulation results for the poor performers PRE, BUM, and SPLOSH. All methods suffer from underestimation of π_0.

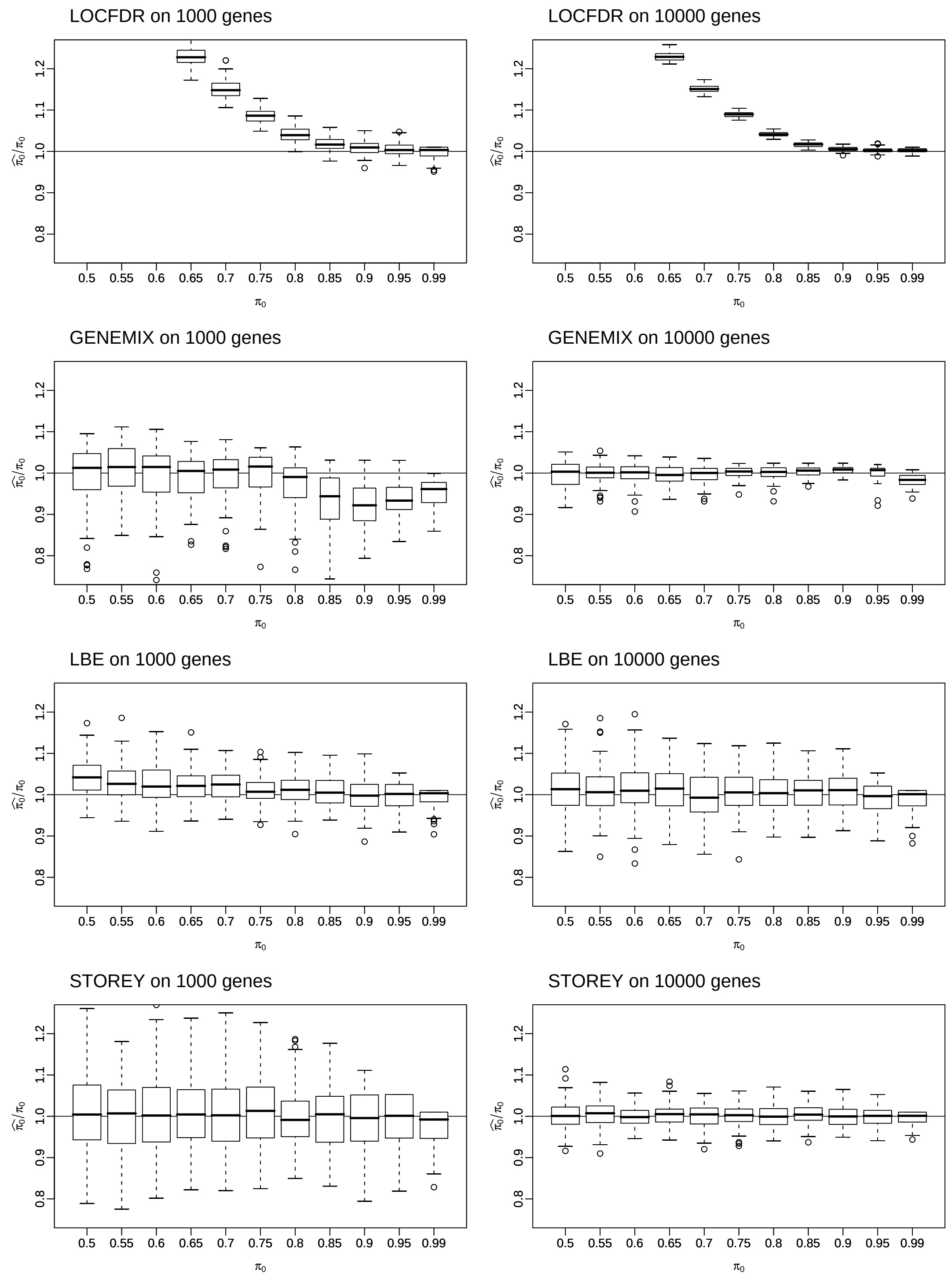

Figure 5.5: Distribution of ratio $\widehat{\pi_0}/\pi_0$ for $\mu = 3$ and varying π_0. Shown are the simulation results for the moderate performers LOCFDR, GENEMIX, LBE and STOREY. The upper bound estimates of LOCFDR are by far too large when π_0 is small. GENEMIX behaves well for large sets of p-values but not for small sets. The variance of the LBE estimates slightly increases for larger sets. STOREY's estimates are unbiased but too variable for small sets.

truly observe $\pi_0 = 0.5$. The GENEMIX approach tends to underestimation if the set of genes is small ($m = 1000$). Surprisingly, it estimates π_0 with high accuracy with the larger sample size of $m = 10000$. A reliable estimator should however yield comparable results under most settings with its major behavior being stable. Additional to the long runtime of GENEMIX, the algorithm did not always converge to a numeric solution. The widths of the boxplots are proportional to the number of observations. A closer look at the GENEMIX plots in Figure 5.5 reveals that the number of successful runs was often smaller than 100.

In contrast to GENEMIX, the variance of the estimates of method LBE increases for higher sample sizes. Method LBE has the highest variance of all compared methods if $m = 10000$. Finally, the approach of STOREY delivers unbiased estimates but shows increased variances. Like SPLOSH and GENEMIX, STOREY's estimates are too variable if the sample size is small.

The good performers. Finally, we are left with seven methods that perform considerably well: LSL, GENOVESE, HOWMANY, NETTLETON, MGF, CONVEST, and SEP. In Figure 5.6 we display the results of the clear upper bound estimators LSL, GENOVESE, and HOWMANY. Methods HOWMANY and GENOVESE perform equally well with HOWMANY showing less variation than GENOVESE. Yet both methods are biased upwards and approach the equality line only for almost uniform p-value samples. The estimates of method LSL are more conservative than those of GENOVESE and HOWMANY. We even observe increasing bias with larger sample size. For $m = 10000$, the two bounding theory based methods GENOVESE and HOWMANY clearly outperform the lowest slope estimator LSL.

In the following, we keep the upper bound estimate of HOWMANY as upper baseline and concentrate on the less conservative methods NETTLETON, MGF, CONVEST, and SEP shown in Figure 5.7. Methods NETTLETON and MGF are slightly biased upwards thus providing estimates that are less conservative than those shown in Figure 5.6. CONVEST and SEP perform well with accurate estimates but higher variation. These four approaches do not provide upper bounds for all parameter combinations and therefore carry the risk of underestimating π_0.

Figures 5.8 and 5.9 provide a comprehensive overview on the competing methods. Shown are the ratios of $\widehat{\pi_0}/\pi_0$ averaged over hundred runs of each parameter combination (π_0, μ) with $m = 10000$. The colors correspond to the ratio intervals depicted in the bottom right panel of Figure 5.8. The brightest non-white

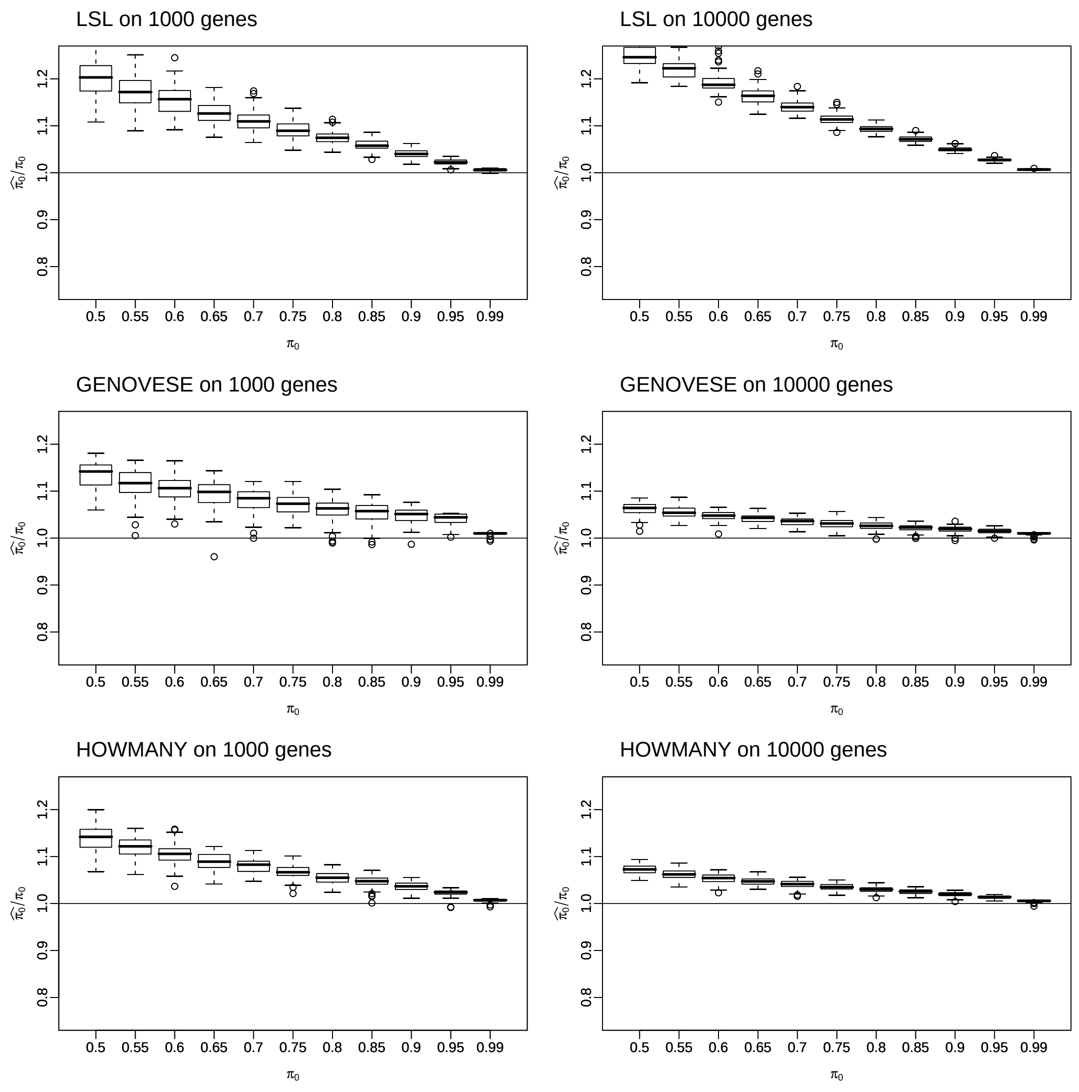

Figure 5.6: Distribution of ratio $\widehat{\pi_0}/\pi_0$ for $\mu = 3$ and varying π_0. Shown are the simulation results for the good upper bound estimators LSL, GENOVESE, and HOWMANY. The methods provide conservative estimates with low variances. However, LSL tends to substantial overestimation with even more conservative estimates for larger sample size.

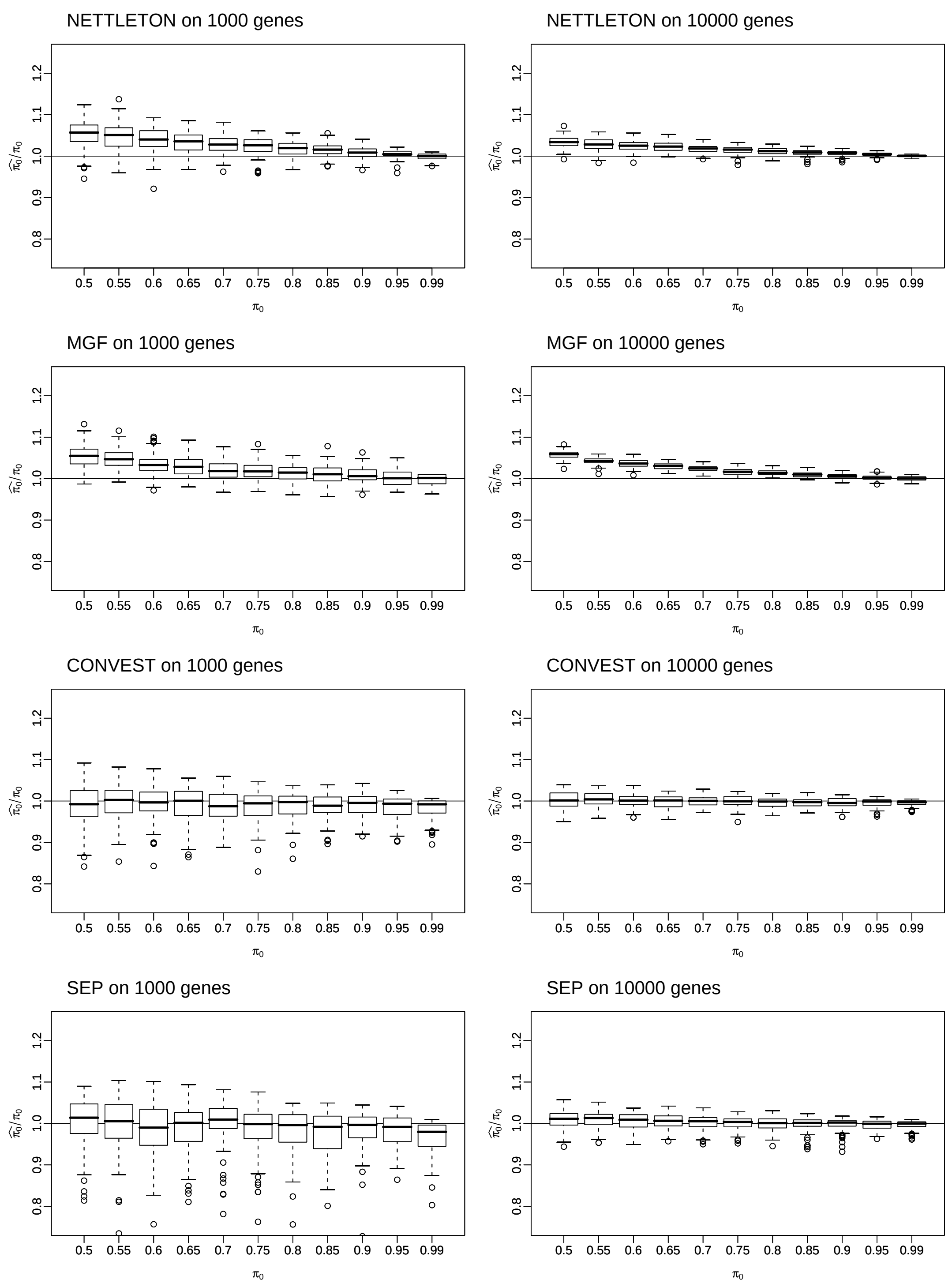

Figure 5.7: Distribution of ratio $\widehat{\pi_0}/\pi_0$ for $\mu = 3$ and varying π_0. Shown are the simulation results for the good performers NETTLETON, MGF, CONVEST, and SEP. MGF and NETTLETON yield similar upper bounds. CONVEST and SEP are less biased but show higher variances than MGF and NETTLETON.

color relates to estimated values close to the target π_0. The equality is marked by thick contour lines. Deeper colors relate to overestimated values. Regions where underestimation occurs are shown in white, starting from ratios $\widehat{\pi_0}/\pi_0$ below 0.9. Thus, a reliable method should not include white regions. The more of the parameter space is covered by the brightest color, the better. Deep colors are only expected in the lower left corners of the plots where the strength of induction μ approaches zero and the mixture distribution appears to be uniform.

The poorly and moderately performing estimators shown in Figure 5.8 do not entirely comply with these requirements. The contour plots resemble the observations above, that is methods PRE, BUM, SPLOSH, and GENEMIX suffer from underestimation while method LOCFDR on the other hand substantially overestimates π_0 even in regions of clear induction signal. The plots of methods LBE and STOREY are comparable to those of the good performers in Figure 5.9. Yet the zigzag line denoting the region of equality between π_0 and its estimates resembles the somewhat enlarged variances of LBE and STOREY, which we observed in Figure 5.5.

The plots change substantially for the well performing methods in Figure 5.9. Within the parameter space, there are no regions of underestimation and the equality contour lines appear as clear bands. The good performers differ with respect to the size of the bright area covering ratios between 0.9 and 1.1. The smallest bright area was observed for method LSL, which was the most conservative of the good performers, followed by the two upper bound estimators GENOVESE and HOWMANY. For the remaining methods we observe more or less continuous contour lines of very accurate estimates with $\widehat{\pi_0}/\pi_0 \in (0.9, 1.1]$.

To provide a final and direct comparison of the good performers, we display mean ratios $\widehat{\pi_0}/\pi_0$ with double standards errors of the different methods in one plot. For better visibility, we connected the mean ratios with lines. The resulting curves are shown in Figure 5.10. The values are drawn over the range of π_0, and the eight subplots differ with respect to μ increasing from 0.5 to 4 in steps of 0.5. From these plots, one can examine both accuracy and variability of the estimates. If μ is high, the two score distributions belonging to induced and non-induced genes separate well. Thus the mixture parameter π_0 is easier to estimate from the resulting p-value distribution for large μ than for small μ. The estimates improve substantially with increasing μ as well as with high percentage π_0. For small μ and small π_0, we observe severe overestimation. With the given simulation setting, the upper

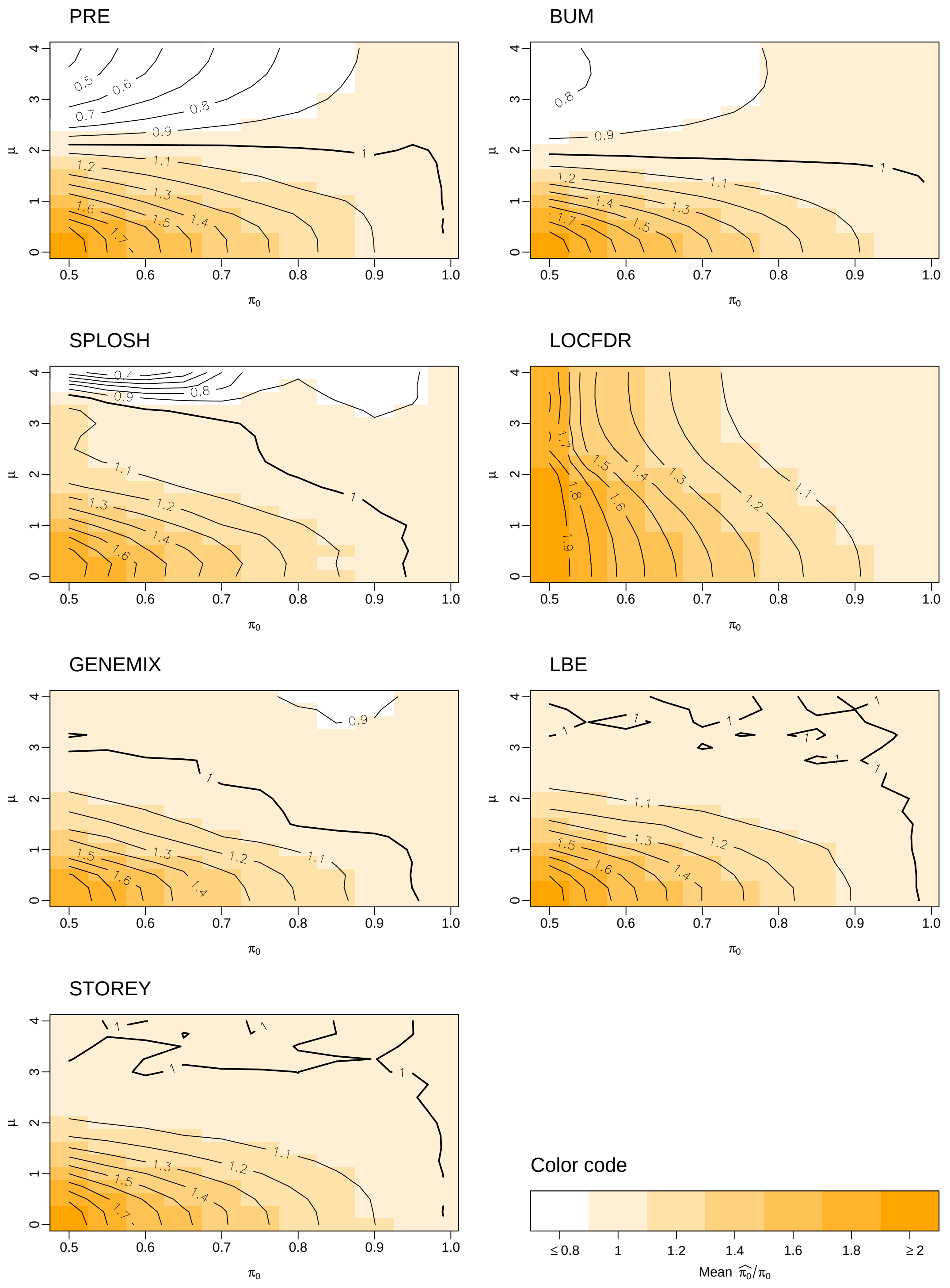

Figure 5.8: Average ratio $\widehat{\pi_0}/\pi_0$ of poor and moderate performers over the parameter space (π_0, μ). Shown are the results for $m = 10000$ genes except for GENEMIX, where due to long runtime the complete parameter space was only evaluated for $m = 1000$. The values are color-coded and overlaid by contour lines. For color code see bottom right panel.

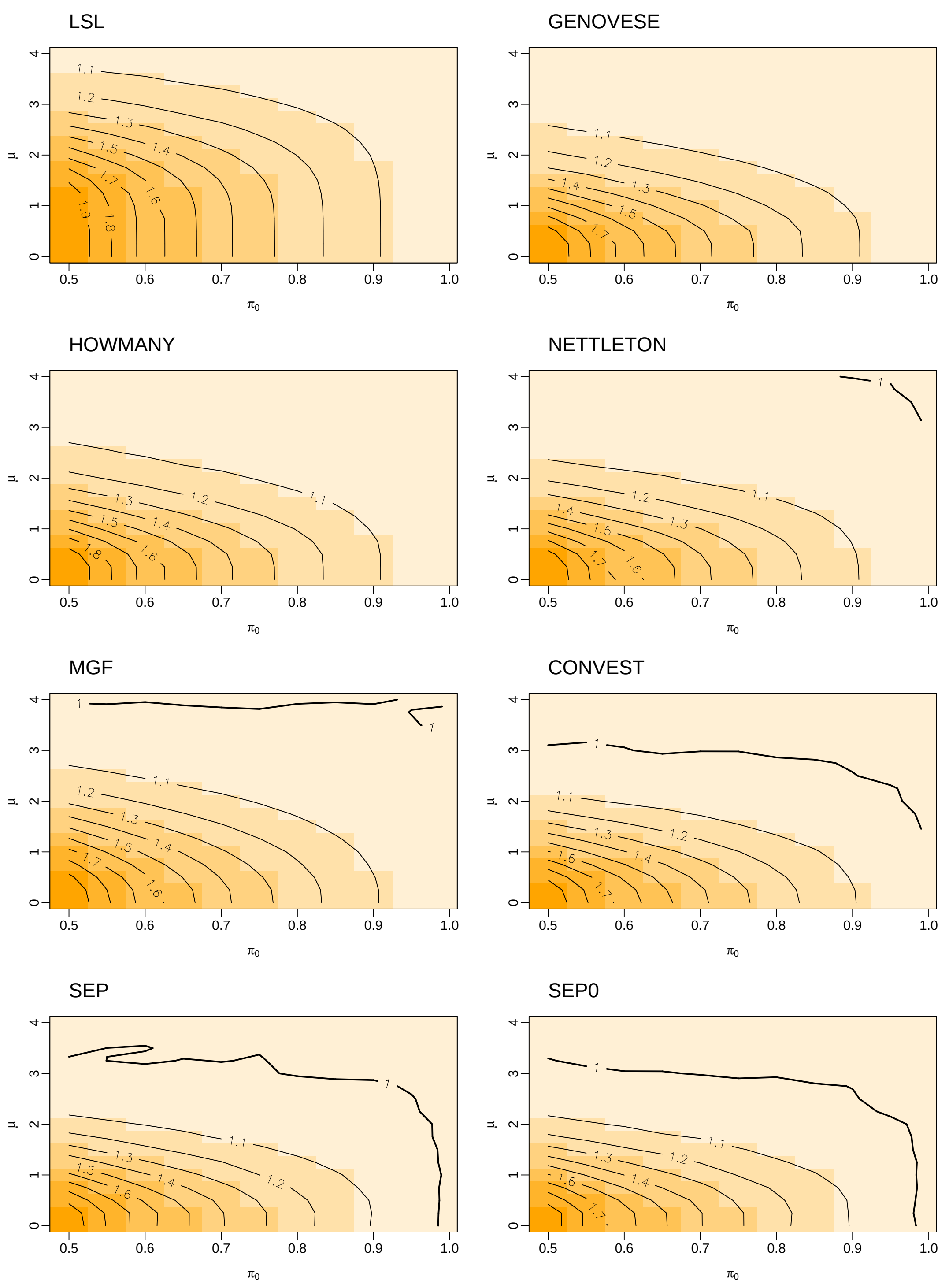

Figure 5.9: Average ratio $\widehat{\pi_0}/\pi_0$ of good performers over the parameter space (π_0, μ). Shown are the results for $m = 10000$ genes. The values are color-coded and overlaid by contour lines. For color code see bottom right panel of Figure 5.8.

bound for overestimated ratios is 2: when μ is small, the two score distributions are not separable any more and every method will return $\widehat{\pi_0}$ close to one. In the worst case of $\mu = 0$ and $\pi_0 = 0.5$, this results in the ratio $\widehat{\pi_0}/\pi_0 = 2$.

The baseline upper bounds of HOWMANY stay well above the equality line while the estimated ratios of the remaining methods approach to one with increasing μ, that is the clearer the two score distributions separate. For low values of μ, MGF (purple line) is the only method that yields estimates above the upper bound if π_0 is small. With increasing μ however, the effect levels off and the estimates drop down to the equality line. The estimates of MGF improve quickly and even outperform NETTLETON (green line) in the end. For $\mu = 4$, MGF tends to slight underestimation. NETTLETON outperforms MGF for low values of μ and stays above the equality line over the whole range. Even for $\mu = 4$, it slightly overestimates π_0 thus being a conservative and reliable approach.

As mentioned above, we included SEP without regularization (SEP0) into the comparison. Surprisingly, although it is prone to underestimate π_0, SEP0 (orange line) is hardly distinguishable from CONVEST (blue line). Both sets of ratios are more variable than those of the previous methods but closer to the equality line. From $\mu = 3$ on, both approaches tend to underestimation. We expected this behavior of SEP0 since it was run without regularization, which otherwise safeguards against too many exclusions from the uniform part. SEP with regularization (red line) performs close to SEP0 and MGF but stays on average above the two estimators. For high μ, it also drops below the equality line.

Summing up the results. It is hard to decide for a single winner. Conditioned on the setting, one would favor one or the other method. For low μ and π_0, method MGF might be least favorable since it yields large upper bound estimates. Yet it improves substantially for higher parameter choices. Method NETTLETON provides conservative estimates. It has the additional advantage of being an analytic estimator and can be used for reliable and quick pre-estimates. Our proposed method SEP performs well with slight underestimation for high μ only. However, neither the strength of induction nor the true π_0 are known in real data. In our opinion, the underestimation is not critical here. For large μ, SEP yields ratios above 0.99 on average which corresponds to estimates ranging from $\widehat{\pi_0} \geq 0.4950$ for $\pi_0 = 0.5$ to $\widehat{\pi_0} \geq 0.9801$ for $\pi_0 = 0.99$. These differences are negligible. Also, the runtime of SEP is not of crucial importance: we refer to the following section for details on the implementation of SEP in package *twilight* and an evaluation on

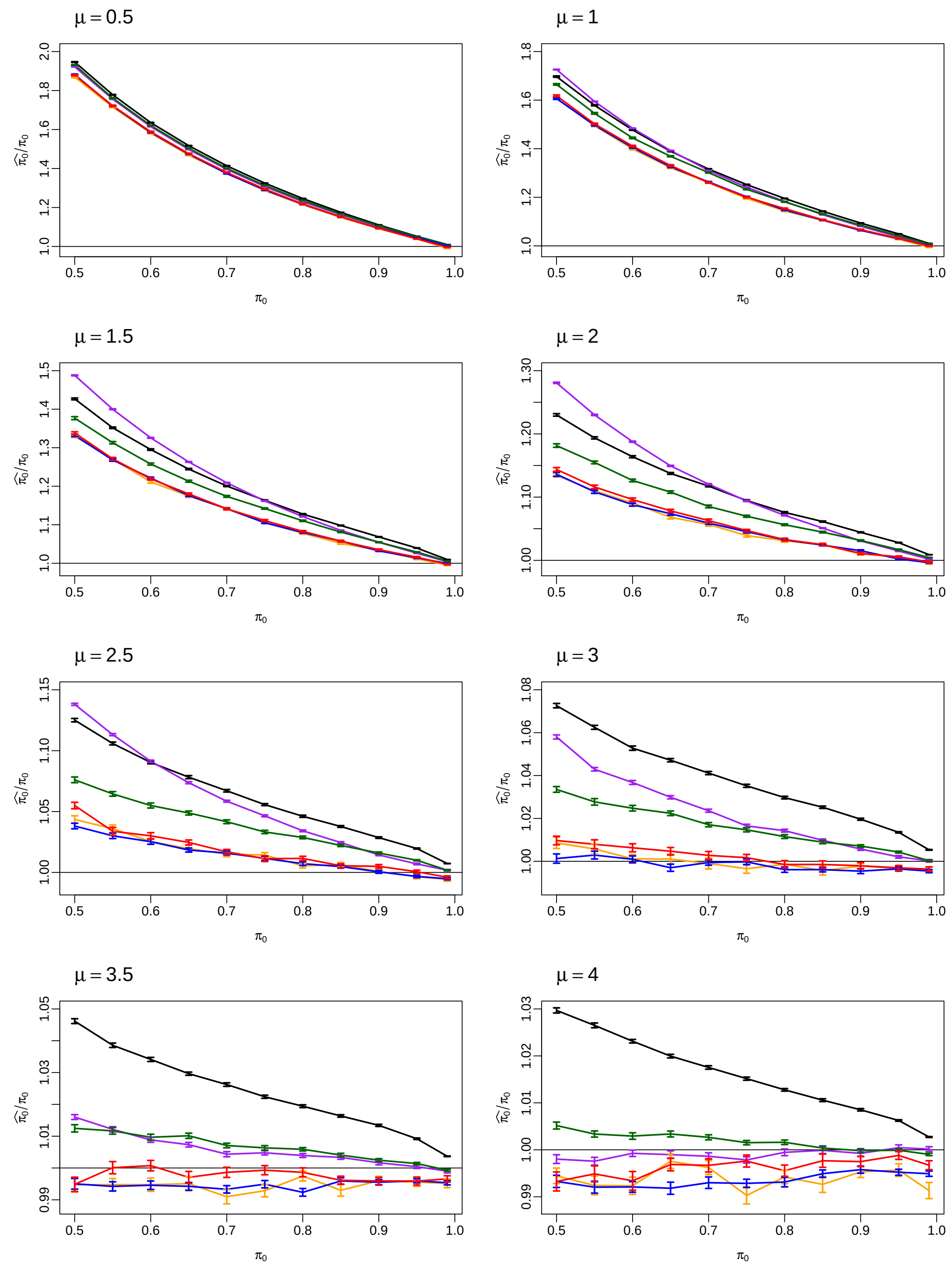

Figure 5.10: Average ratios of $\widehat{\pi_0}/\pi_0$ with standard errors plotted for different strengths of induction μ. The estimates are based on $m = 10000$ genes. Note the different scales of the y-axes. The colors correspond to the following methods: HOWMANY (black), MGF (purple), NETTLETON (green), CONVEST (blue), SEP0 (orange), and SEP (red).

runtime. Our final favorites are Sep for providing reliable estimates consistently for all parameter settings, and Nettleton for providing fast and conservative estimates.

5.7 Implementation and runtime evaluation

We close with a software chapter introducing our package *twilight*, in which we implemented the algorithms as defined above. The package is called *twilight* as its key function outputs SEP estimates of the local false discovery rate: following the local false discovery rate curve over the range of p-values reveals possible twilight zones where evidence for differential expression levels off into non-differential expression. The implementation supports the following analysis layout: the user chooses a scoring method and possible parameters like number of permutations for computing empirical p-values and number of bootstrap samples for estimating the local false discovery rate. In the following, we introduce the key functions of *twilight* and discuss findings of a runtime evaluation. In addition, the package provides plotting functions to explore the results.

Implementation. The package is written in the statistical software language R (R Development Core Team, 2005) and is distributed through the Bioconductor Project, a collection of software for genomic data analysis (Gentleman *et al.*, 2004). The package itself divides into two major parts. First, it offers fast means of computing scores for large microarray data sets. The scoring methods are limited to paired and unpaired data with patients belonging to two clinical classes, and to scores assessing the correlation to a clinical variable. We implemented three two-class scores: the difference of class averages or log ratio score defined in Equation (2.4), the t-score defined in Equation (2.8) and the z-score defined in Equation (2.10). The fudge factor for the z-score might be chosen by the user. Otherwise, it is set to the median of the pooled standard deviations across genes as defined in Equation (2.9). Linear correlation is measured with the common Pearson's or Spearman's coefficients of correlation.

The scores are computed for each gene and transformed to p-values by applying the permutation scheme of Equation (2.14). The user might choose to input an individual set of permuted class labels, otherwise permutations are drawn randomly. If the sample size and hence the number of possible permutations are small, the software proceeds with a complete enumeration of all possible permutations. From the p-values we estimate the percentage of non-induced genes π_0 and the positive false discovery rate by computing q-values as given in Remark B of Storey and Tibshirani (2003). Finally, we apply the SEP procedure as given in Tables 5.1 to 5.4 to estimate the local false discovery rate from the set of p-values, possibly accompanied by bootstrap estimates and confidence intervals. A valuable by-product are bootstrap estimates and confidence intervals for prior π_0. So far, the q-values were calculated with the π_0 estimator of Storey and Tibshirani (2003). After estimation of the local false discovery rate, the q-values get updated with the SEP estimate $\widehat{\pi_0}$. The user might explore the final output using a summary function and several implemented plots. For example, we can plot the local false discovery rate estimates together with bootstrap confidence intervals over the entire range of p-values as was done in Figure 5.3. Another possibility is the choice of an adequate positive false discovery rate threshold by exploring the size of the resulting gene lists similar to Figure 4.3.

Computational remarks. The expensive parts of *twilight* are written in C and called internally from R. Thus the computation of permutation-based scores and p-values and the estimation of the local false discovery rate are fast. To keep the runtime for the bootstrapping part low, the use of a Linux cluster at hand is possible. The bootstrap operations will be distributed on the cluster using the functionality of package *snow* by L. Tierney, A.J. Rossini, N. Li, and H. Sevcikova.

Runtime evaluation. We proceed with a runtime evaluation of the core SEP implementation. The algorithm starts with the full set of input p-values and successively removes and adds some until the remaining set is close enough to a uniform sample. In each step, one p-value is selected for removal or inclusion, and the objective function is evaluated regarding the new configuration. One might expect that the required time for one run of SEP increases the more the original p-value sample deviates from uniformity. To test this behavior, we return to our common simulation setting as introduced in Section 5.6. For $m = 1000$, 2000 and 3000 genes, we draw random test scores from two normal distributions, namely $\mathcal{N}(0,1)$ and $\mathcal{N}(\mu,1)$ for non-induced and induced genes respectively, and transform the scores to p-values under the null distribution $\mathcal{N}(0,1)$. The strength of induction is set to $\mu = 3$. The amount of non-induced genes π_0 varies over the whole interval $(0,1)$. Note that in the previous simulation in Section 5.6, we only examined $\pi_0 \geq 0.5$.

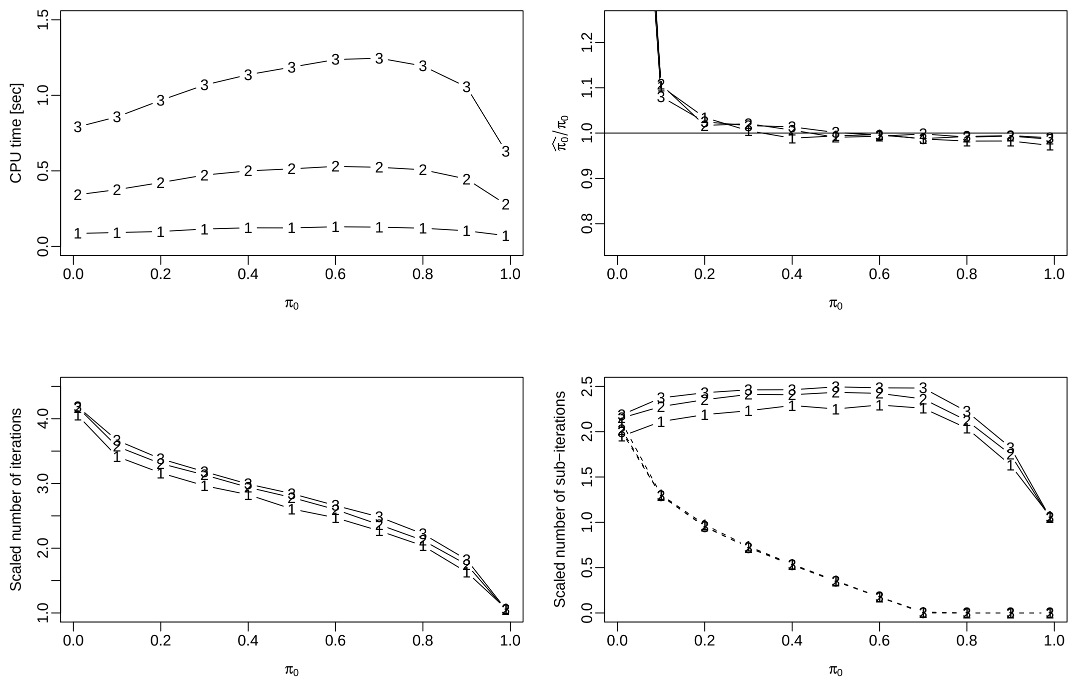

Figure 5.11: Runtime evaluation of SEP for varying numbers of genes m and increasing percentages π_0 of non-induced genes. Top: CPU time used per run, and estimated ratio $\widehat{\pi_0}/\pi_0$. Bottom: Total number of internal iterations scaled by the minimum number $2m$, and scaled numbers of internal subroutines, that is initial iteration without regularization (dashed lines) and successive iteration with regularization (solid lines). Values are averaged over 100 runs of SEP. Annotation "1" corresponds to $m = 1000$, "2" to $m = 2000$ and "3" to $m = 3000$.

For each combination (π_0, m) we run 100 repeats of SEP. For an unbiased comparison, we broke SEP down to the source code written in C. The function performs one run of SEP without regularization and outputs a binary vector of length m indicating the two p-value sets after separation into an induced and a non-induced part. We extended this function to count and output the number of iterations the algorithm needed internally. The time was measured for each single run of the C function called from R. Additional computation, like calculating the final ratios $\widehat{\pi_0}/\pi_0$, were done subsequently and did not contribute to the time measurement. The simulation was carried out on a Linux machine with AMD Athlon™ 3200+ CPU.

In Figure 5.11, we show the results conditioned on π_0. We measured three parameters: the CPU time used for the core SEP, the number of iterations within SEP, and the accuracy of the final estimate. The top right plot in Figure 5.11 is equivalent to those in Section 5.6, displaying the ratio $\widehat{\pi_0}/\pi_0$ condensed to means instead of boxplots. The SEP algorithm estimates the true values well. For lower π_0, we observe overestimation reflected in high ratios due to low values π_0. For higher π_0, SEP underestimates the true values slightly since we applied it without the safe-guard of regularization.

The bottom left plot shows the number of iterations scaled by the minimum number of needed iterations. Since the SEP algorithm stops after twice the number of genes of unsuccessful trials, the minimum number of iterations is $2m$. In the top left plot of Figure 5.11 we depict the CPU time measured in seconds. Surprisingly, the needed time does not increase with decreasing amount π_0 of non-induced genes, that is uniformly distributed p-values, but decreases after a maxima around $\pi_0 = 0.7$. In contrast to this, the number of iterations increases with decreasing percentage of uniform p-values. This behavior is consistent to our expectations since the number of values to be removed increases with decreasing amount of uniformly distributed values. It is not consistent to the parabola shape of the CPU time above since we expect many iterations to last longer than a few.

The discrepancy is easy to explain: it is triggered by the algorithm fine-tuning introduced in Section 5.4. The SEP algorithm is divided into two parts: depending on the Kolmogoroff-Smirnoff (KS) test statistic, we either run SEP once with regularization if the distribution is already close to uniform, or once without and once with regularization if we have to remove many genes before the regularized fine-tuning can begin. The critical point is a KS value of 0.25, see Table 5.4. We evaluated the simulation data and observed that $\pi_0 \geq 0.7$ yields p-value distributions with KS ≤ 0.25. Hence, up to $\pi_0 < 0.7$, SEP runs twice, and only once if $\pi_0 \geq 0.7$. For further evaluation, we measured the iterations needed within the two subroutines separately. The result is shown in the bottom right plot of Figure 5.11. The dashed lines are the scaled numbers of iterations of the first run without regularization. From $\pi_0 = 0.7$ on, the first round is skipped due the KS value, and the number of iterations equals zero. The solid lines correspond to the second run with regularization. We observe that the number of iterations stays almost constant up to $\pi_0 = 0.7$. Since we actually run SEP without any regularization the second round is not regularized and does not break too early although we have already removed more values than allowed.

However, the numbers of subroutine iterations do not explain the parabola curve of the CPU time. The effect source is more subtle: in each iteration we have to evaluate the goodness-of-fit to the uniform distribution using the KS test. The time needed to compute the KS statistic depends on the number of observations under test. Hence, due to the removal of p-values in the first round, the KS evaluations in the second round are faster for smaller values of π_0 than for larger. The effect levels off with increasing π_0 and becomes compensated because few iterations are still faster than many. For high values of π_0, the initial distribution is close to uniform, and the run iterations hardly exceed the number of unsuccessful trials $2m$, which leads to scaled iteration numbers approaching one and to the drop in CPU time.

The core SEP algorithm is fast and thus of marginal contribution to the total runtime of package *twilight*. The wrapping function performs data checks and stores the output data in a comprehensible format. The total runtime depends on the number of genes and the strength of regularization. A regularized run without bootstraps on the complete exemplary data sets is done within a few minutes each. We argue, that—apart from its functionality—package *twilight* offers fast means of computation and is suitable for every-day's data analysis.