2 Introduction to microarray data

2.1 Outline

We start by introducing the basics of microarray experiments, which provide the data underlying this thesis. The raw data output of microarray experiments needs several adjustments, the so called preprocessing steps. In Section 2.3 we describe the successive steps that were applied to the data sets within the thesis. After preprocessing, Section 2.4 formally describes the derivation of the fundamental ingredients for microarray analysis: scores and p-values. The p-values form the basis for further analysis of significance, in particular for the estimation of the global and local false discovery rate, which we will review in Chapter 3.

2.2 Microarray data

Microarrays provide snapshots of cell activity. With a microarray device it is possible to observe the signal transduction between the genes, which store the building plan for the assembly of proteins, and the cell compartments where proteins are built. The signalling device is called mRNA, a molecule, which resembles the information stored in a gene. The more copies of a protein are needed by the cell, the more copies of the corresponding mRNA molecule are built. In principle, the amount of specific mRNA copies refers to the activity of the associated gene. The more copies there are, the higher the gene is *expressed*, and the more of its corresponding protein will be assembled. Genes with high abundance of mRNA copies are called *up-regulated* with respect to some commonly observed abundance. If there are no or only a few specific mRNA copies present, the associated gene is called *down-regulated*. Up- and down-regulation is summarized in

the term *induction*: induced genes are either up- or down-regulated. Genes that are not induced are called *consistently expressed*.

Microarrays are micro chips measuring gene expression. The array surface is divided into small areas called *probes*. Each probe holds one type of mRNA molecule—or rather one uniquely characterized subunit of that mRNA molecule. The probe contains molecules complementary to the mRNA molecule. If one copy of our mRNA sample reaches its complement on the array, the two molecules bind with each other and the mRNA copy stays fixed at this probe. This procedure is called *hybridization*. A microarray allows to measure the expression of genes simultaneously. To this end, all mRNA molecules are extracted from the cells in a given tissue sample and, after additional amplification steps, washed over the array surface. The mRNA molecules hybridize with the complements on their specific probes.

Prior to hybridization, each of the amplified mRNA fragments was labeled with a dye molecule. The dye molecules are excited by laser light such that every mRNA fragment, which hybridized to a probe, emits a light signal. Each probe contains several millions of complement copies to allow for a quantitative measurement: probes containing many hybridized mRNA molecules emit more light and are thus brighter than probes with less hybridizations. Bright probes correspond to high mRNA abundance and dark probes correspond to low abundance. The light emission is read out by a scanning device. To this end, the array surface is divided into pixels and the intensity values are measured per pixel. An example of a scanner read-out is shown in Figure 2.1. The raw intensity data need several preprocessing steps before we can analyze the final expression values. The preprocessing procedures will be introduced in Section 2.3. So long, we assume for simplicity the data to be preprocessed such that we are given one expression value per gene.

The result of one microarray experiment are expression values of one individual tissue sample. To explore a second tissue sample, a new chip has to be used. Thus a *microarray data set* consists of data from several single microarrays. A data set might comprise measurements of independent tissue samples or repeated measurements of the same sample like in a time-series experiment. In the following, we restrict the analysis to samples derived from independent patients. Each patient can be clearly categorized with respect to a disease status. For simplicity, we assume that we only observe patients of two distinct disease classes. Each patient

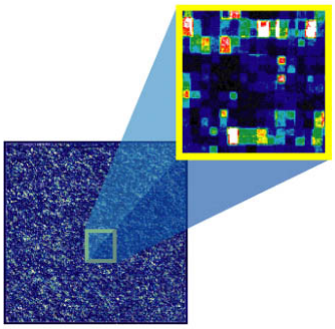

Figure 2.1: Intensity values measured on a microarray. The intensity values per pixel are read out by a laser scanner. A probe of high intensity provides evidence for a highly expressed gene. Light colors correspond to high abundance, dark colors to low abundance. Single probes appear as rectangular subunits. Image courtesy of Affymetrix®.

belongs to one of these two classes. The classical use of microarrays is to compare the gene expression of two disease classes and we will introduce several two-class data sets in Chapter 4, which serve as illustrative examples of the introduced methods. The goal is to identify genes showing different expression between the two disease classes. This might lead to a better understanding of the disease and to possible conclusions regarding progression or even cure of the disease.

2.3 Preprocessing microarray data

We proceed with a technical description of the applied preprocessing steps for microarray data leading from the raw scanner read-out as shown in Figure 2.1 to the expression data matrix $\mathbf{X}$, on which we base our analysis. In Chapter 4, we introduce six data sets all measured on the same microarray platform. Here we describe the platform and the preprocessing methods we used.

A closer look at Affymetrix® arrays. The microarray used for the six data sets in Chapter 4 is the Affymetrix GeneChip® HGU95Av2 array. The HGU95Av2 measures the mRNA of 12625 genes in total. The array surface of about 2 cm^2 is divided into probes, each one contains more than 100 million copies of its specific mRNA molecule. More precisely, each probe is specific to a *fragment* of its gene's mRNA and each gene is measured not only by one probe but by a set of 16–20 probes. Each of these probes is specific to a different fragment of the gene's mRNA. The summary expression over these probes gives a reliable measurement

of gene expression. The molecules on the probes are exact complements to the mRNA fragments and are thus called *perfect match probes*. Along with the perfect match probes comes the same number of *mismatch probes*. A mismatch probe is almost identical to its perfect match probe. However, the specific mRNA fragment does not hybridize on the mismatch probe. Mismatch probes are assumed to reflect nonspecific hybridization and are used to model the background expression. The set of perfect and mismatch probes encoding for one gene is called *probe set*. The probes belonging to one probe set are randomly distributed on the surface grid to avoid loss of a complete set due to regional hybridization errors.

Image analysis. The initial steps of preprocessing, that is image analysis and background correction, are done similar as given in the Affymetrix® software Microarray Suite 5.0 (MAS 5.0) specifications (Affymetrix, 2001, 2002). The readout scanner image of a microarray consists of a pixel representation of the light intensities on the chip surface, see Figure 2.1. The first preprocessing step is the recognition of the original grid on which the probes were arranged. Each probe is then represented by a squared area of pixels. The following steps are applied to reduce the different values to one intensity value per probe: first the framing pixels of each probe are omitted as the border between two neighboring probes is often hard to define. The remaining pixels are then averaged into one value per probe.

Background correction. Each microarray emits a certain background intensity. This intensity offset must be subtracted from the probe intensities. For calculating the background intensities, the array surface is divided into a grid of 4×4 squares. In each square, the average of the lowest 2% probe intensities is taken as background measure. Next we compute an individual background value for each probe in each square: for a single probe, the distance of the probe to the *center* of each of the 16 squares is calculated. The probe's background is then computed as a weighted average of the squares' background values with weights being proportional to the reciprocal squared distances. The distances are illustrated as arrows in Figure 2.2. The individual background is then subtracted from the probe's intensity. Note that the original procedure of the Affymetrix® MAS 5.0 software includes more adjustments to avoid negative intensities after background correction. We do not use these corrections as the successive methods are capable of dealing with negative intensities.

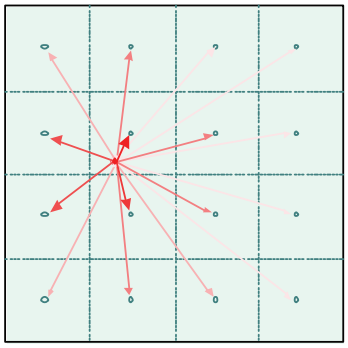

Figure 2.2: Affymetrix® MAS 5.0 background correction. The array surface is divided into 16 squared areas. An averaged background value is assigned to the center of each square. For each probe, the distances of the probe to the area centers are computed. The local background value is then computed as the weighted average of the squares' background values with weights being proportional to the reciprocal squared distances. The weights are indicated by color strength. Image courtesy of Affymetrix®.

Normalization. So far, we corrected each array individually. The next preprocessing step is to normalize *between* arrays. Although they are background corrected, the mean intensity of the arrays can still vary due to different laser calibrations or different experimental conditions. To compare several arrays, the intensities have to be normalized to remove unwanted confounders. We used a variance-stabilizing procedure of Huber *et al.* (2002) implemented in package *vsn* by W. Huber. Besides normalization between arrays, variance-stabilization removes the unwanted effect that high intensity values show larger variation across arrays than low values. Variance-stabilization returns intensity values with variation being independent of mean intensity. We briefly review the procedure as given in Huber *et al.* (2002, 2003). Let $\mathbf{Y}$ be the matrix of measured probe intensity values with y_{kj} being the intensity of the kth probe and the jth array (patient). Standardizing the intensity values by a patient offset a_j and a patient scale factor b_j results in $(y_{kj} - a_j)/b_j$. A variance-stabilizing transformation is obtained by choosing the inverse hyperbolic sine function

$$\operatorname{arsinh}(x) = \log\left(x + \sqrt{x^2 + 1}\right). \tag{2.1}$$

The transformation is applied to the standardized values and the result is further decomposed into

$$\operatorname{arsinh}\left(\frac{y_{kj} - a_j}{b_j}\right) = \mu_{kj} + \varepsilon_{kj}, \tag{2.2}$$

with μ_{kj} being the non-confounded intensity on transformed scale and ε_{kj} being a normally distributed error term for which $\varepsilon_{kj} \overset{\text{iid}}{\sim} \mathcal{N}(0, \sigma^2)$. The transformation parameters are estimated using a robust version of maximum likelihood estimation. The procedure returns variance-stabilized intensity values on the arsinh scale. This scale is additive in the sense that for highly expressed genes differences in normalized values correspond to fold changes in original values. For highly expressed genes the inverse hyperbolic sine approximates the logarithm plus a constant. For lower expressed genes, log and arsinh differ substantially as the arsinh behaves linearly in the lower range. Most statistical applications assume additivity and hence the mapping onto an additive scale is necessary to proceed with the analysis.

Summarization. The final preprocessing step is the conversion of the normalized probe intensity matrix $\mathbf{Y}$ into the gene expression matrix $\mathbf{X}$. We need to summarize the 16–20 probes with perfect match values associated to the same gene into one final expression value. To this end we apply a robust median-polish method introduced by Irizarry *et al.* (2003a). Let y_{ijl} be the normalized intensity of the lth probe belonging to the probe set of the ith gene for patient j. For each probe set, the values are decomposed into overall offset μ, probe effect α_i and patient effect β_j yielding

$$y_{ijl} = \mu + \alpha_i + \beta_j + \varepsilon_{ijl}, \tag{2.3}$$

with error terms $\varepsilon_{ijl} \overset{\text{iid}}{\sim} \mathcal{N}(0, \sigma^2)$. The model is fitted using a median-polish approach. The final summarized expression values are then given as the estimated offset plus patient effect. For a review and comparison of preprocessing methods and quality control of microarray data we refer to Bolstad *et al.* (2003) and Irizarry *et al.* (2003b).

2.4 Assessing differential gene expression

We assume the preprocessed gene expression data to be stored in a matrix $\mathbf{X}$ with entry x_{ij} being the expression value of the ith gene and the jth patient. Gene expression was measured for m genes and a panel of n independent patients such that matrix $\mathbf{X}$ is of dimension $m \times n$. Each patient belongs to one of two distinct

disease classes. The class labels are stored in a vector $\boldsymbol{c}_0 = (c_1, \ldots, c_n)$ with c_j being the label of the jth patient. Our goal is to compare the two disease classes, which relates to finding differences in gene expression. For each gene we might ask: does the expression differ between the two classes? If it does, the gene is called *differentially expressed*. In general we assume genes *not* to be differentially expressed. In statistical terms, we introduce a null hypothesis H_0:

H_0: The gene is not differentially expressed between the two classes.

The null hypothesis states that there is no significant difference in the expression values with respect to the classes. If we reject the null hypothesis, the gene is said to be induced. Induction is further distinguished into up- and down-regulation, that is we observe either an increase or a decrease of gene expression. Up- and down-regulation is always given with respect to one of the two classes: if gene i is up-regulated in the first class, it is down-regulated in the second class. We further refine the null hypothesis above into a simpler version, that is we reduce the term "differentially expressed" to "difference in mean expression":

H_0: The gene has equal mean expression in the two classes.

A score for differential expression. The first step in the significance analysis of gene expression data is to provide a ranking of the genes with respect to differential expression. For each gene i, $i = 1, \ldots, m$, we compute a score s_{i0} quantifying its differential expression. We assume that a high positive score corresponds to up-regulation and a high negative score to down-regulation. A simple score that complies with this assumption is the log ratio score, which is the difference in expression means of the two classes. The term *log ratio* refers to the popular *fold change* measure: microarray data are originally on a multiplicative scale, that is a gene is said to be differentially expressed if its mean expression is for example twice as high in the first class than it is in the second class. As most statistical methods work on additive scale, one preprocessing step is the log-transformation of the data. Then the difference between values on logarithmic scale corresponds to the fold change ratio on original scale and is thus called log ratio. Our preprocessed data complies with the additive scale requirement as the variance-stabilization introduced in the previous section returns values on additive scale.

If we assume the class labels to be binary, the log ratio score s_{i0} for gene i is given as

$$s_{i0} = \bar{x}_{i\{c=1\}} - \bar{x}_{i\{c=0\}}, \tag{2.4}$$

where $\bar{x}_{i\{c=1\}}$ and $\bar{x}_{i\{c=0\}}$ are the mean expression values for class 1 and 0, respectively, that is

$$\bar{x}_{i\{c=1\}} = \frac{1}{n_1} \sum_{j=1}^{n} x_{ij} I\{c_j = 1\} \quad \text{and} \tag{2.5}$$

$$\bar{x}_{i\{c=0\}} = \frac{1}{n_0} \sum_{j=1}^{n} x_{ij} I\{c_j = 0\}, \tag{2.6}$$

with $I\{y\}$ being an indicator function that returns 1 if y is true and 0 otherwise. The factors n_1 and n_0 denote the numbers of samples per class:

$$n_1 = \sum_{j=1}^{n} I\{c_j = 1\} \quad \text{and} \quad n_0 = \sum_{j=1}^{n} I\{c_j = 0\}. \tag{2.7}$$

A gene with a high average expression in class 1 and lower average expression in class 0 has a high log ratio score. However, the log ratio score assesses the difference in mean expression but does not incorporate how closely the values are spread around their mean values. A classical score combining mean difference and variation is Student's t-test. The t-score relates to the log ratio score divided by an estimate of the pooled standard deviation $\hat{s}_i$ of gene i assuming equal variances in the two classes. The t-score is defined as

$$s_{i0} = \frac{\bar{x}_{i\{c=1\}} - \bar{x}_{i\{c=0\}}}{\hat{s}_i}. \tag{2.8}$$

The pooled standard deviation is derived from the two within-class standard deviations:

$$\hat{s}_i = \sqrt{\left(\frac{1}{n_1} + \frac{1}{n_0}\right) \frac{\sum_{j=1}^{n} I\{c_j = 1\}(x_{ij} - \bar{x}_{i\{c=1\}})^2 + \sum_{j=1}^{n} I\{c_j = 0\}(x_{ij} - \bar{x}_{i\{c=0\}})^2}{n_1 + n_0 - 2}}. \tag{2.9}$$

Genes get high t-scores if their values differ on average between the classes but do not show much variation in each class. This is the natural behavior of the t-score, yet it might result in misleading rankings of the genes. Consider a gene with expression values being almost constant in each class with a small difference between the classes. Since we divide the mean difference by a very small estimated standard deviation $\hat{s}_i$, the gene will receive a high score. This leads to problems since the variance of the estimated standard deviation is higher than the variance

of the estimated mean difference. We might substantially underestimate the standard deviation of a gene, thus yielding a mixed up ranking. To safeguard against the effect of underestimated standard deviation, Efron *et al.* (2001) introduced a *regulated* t-score, called z-score, where the denominator is enlarged by a value $s^\star$ called the *fudge factor* such that

$$s_{i0} = \frac{\bar{x}_{i\{c=1\}} - \bar{x}_{i\{c=0\}}}{\hat{s}_i + s^\star}. \tag{2.10}$$

The fudge factor is the same for all genes $i = 1, \ldots, m$ and is commonly chosen from the set of pooled standard deviations $\hat{s}_1, \ldots, \hat{s}_m$. Efron *et al.* (2001) chose $s^\star$ as a quantile of $\hat{s}_1, \ldots, \hat{s}_m$. The z-score is situated between the log ratio score and the t-score: in contrast to the log ratio score, the z-score puts less weight on the mean difference by considering the variance as well but does not suffer from small variances like the t-score.

Throughout the thesis, we will use the z-score for assessing differential expression. The fudge factor is set to the median of $\hat{s}_1, \ldots, \hat{s}_m$. Yet there exist many other scoring methods and we will briefly review them in the following. Besides the original work of Efron *et al.* (2001), Tusher *et al.* (2001) implemented a procedure to choose the fudge factor automatically. The software is called SAM for Significance Analysis of Microarrays. Smyth (2004) "borrows information across genes" by introducing another version of a regulated t-score, the *moderated* t-score, where small variances are raised and large variances are shrunken towards a common value. The intention to protect against small variances is adapted by other authors such as Cui *et al.* (2005), Jain *et al.* (2003), or Wu (2005). Besides these methods where distributional assumptions are needed, scores based on ranked expression values are popular as well. The Wilcoxon ranksum score is the classical rank-based equivalent of the t-score. It assesses the difference in median expression while protecting against outlying values. In the context of gene expression data, there exists a variety of rank-based scores such as those introduced in Lee *et al.* (2005), Martin *et al.* (2004), Neuhäuser and Lam (2004), or Zhao and Pan (2003). Another rank-based approach for microarray data is the pAUC-score, which does not evaluate the difference of single location parameters like mean or median but the *separability*, which is the amount of overlap of the two expression distributions (Pepe *et al.*, 2003). Comparative reviews on the use of

the above scores for microarray data can be found in Broberg (2003), Kooperberg *et al.* (2002), or Pan (2002).

The p-value quantifies the significance of a score. The significance of an observed score is assessed by transforming it into a p-value. The p-value is defined as the probability of observing even higher scores simply by chance, that is the null hypothesis is true and there is no difference in means, yet the score is high due to random fluctuation. With score S being a random variable, the p-value p_i of the ith gene is defined as

$$p_i := Pr\left[|S| \geq |s_{i0}| \,\middle|\, H = 0\right], \tag{2.11}$$

where H is a Bernoulli random variable that is 0 if the null hypothesis is true and 1 otherwise.

From single genes to many genes. The p-value of an observed score is computed from the null distribution of the scoring method. For a t-score, the null distribution is the t-distribution with appropriate degrees of freedom. However, the null distribution is not known for every scoring method. Thus it is commonly recommended to use a *permutation approach* (Dudoit *et al.*, 2002). The permutation approach offers an important advantage. So far, we assessed the significance of a single score. In a microarray experiment, the significance of thousands of scores has to be assessed in parallel. As genes are co-regulated and act together in pathways, the expression data is highly correlated and the underlying correlation structure is unknown. The permutation approach preserves the correlation structure and returns an *empirical*, that is data-driven, p-value. In the following, we review the permutation approach to compute p-values empirically.

Recall from the beginning of this section that $\boldsymbol{c}_0$ is the vector of observed class labels and s_{i0} the score of the ith gene with respect to $\boldsymbol{c}_0$. Let $\boldsymbol{s}_0$ denote the vector of scores with entries $(s_{i0})_{i=1,\ldots,m}$. Let $\boldsymbol{c}$ be a random permutation of the entries of vector $\boldsymbol{c}_0$. Note that we shuffle the class labels but keep the order of genes within patients fixed to preserve the correlation structure between genes. We recompute the score of each gene based on $\boldsymbol{c}$ and derive a set of scores $\boldsymbol{s}$. Now we permute the labels in $\boldsymbol{c}_0$ B times, which results in B random label vectors $\boldsymbol{c}_1, \ldots, \boldsymbol{c}_B$. From those we yield B random score vectors $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_B$. We join the original and the random

scores into the $m \times (B+1)$ score matrix $\mathbf{S}$ defined as

$$\mathbf{S} := (\boldsymbol{s}_0\, \boldsymbol{s}_1 \cdots \boldsymbol{s}_B) = (s_{ib}) \text{ with } i = 1, \ldots, m \text{ and } b = 0, \ldots, B. \tag{2.12}$$

To compute the empirical p-value for score s_{i0}, we count how often a random score exceeds the observed score of the gene of interest, that is

$$p_i = \frac{1}{m(B+1)} \sum_{k=1}^{m} \sum_{b=0}^{B} I\{|s_{kb}| \geq |s_{i0}|\}. \tag{2.13}$$

For simplicity of notation, we summarize the whole process in a function $U_{\mathcal{C}}$, which maps a fixed vector of class labels $\boldsymbol{c}_0$ to the vector $\boldsymbol{p}_0 = (p_i)_{i=1,\ldots,m}$ of associate p-values, that is

$$U_{\mathcal{C}}(\boldsymbol{c}_0) = \boldsymbol{p}_0, \tag{2.14}$$

where $\mathcal{C} = \{\boldsymbol{c}_0, \boldsymbol{c}_1, \ldots, \boldsymbol{c}_B\}$ is the set of permutations on which we assess the significance of the scores. Equation (2.14) serves as a simplified description of how to compute empirical p-values.

A remark on pooled p-values. The p-values in Equation (2.13) are *pooled* p-values since we pool across genes thus assuming that all genes follow the same null distribution of scores. A second way of computing p-values is to use *gene-wise* p-values. Here the p-value of gene i is only based on the null distribution estimated from its permutation scores $(s_{ib})_{b=0,\ldots,B}$. The gene-wise p-value $p_i^\star$ of gene i is defined as:

$$p_i^\star = \frac{1}{B+1} \sum_{b=0}^{B} I\{|s_{ib}| \geq |s_{i0}|\}, \tag{2.15}$$

counting how often a permutation score of gene i absolutely exceeded the observed score s_{i0}. Using gene-wise p-values, we assume that genes do not have equal null distributions of scores. Both p-value methods are used in microarray literature. For example, pooled p-values are used in Storey and Tibshirani (2003), while Dudoit *et al.* (2002) support gene-wise p-values. In a slightly different way, pooling across genes is used to assess significance in Tusher *et al.* (2001). Here random scores are pooled to estimate order statistics under randomization. In Ge *et al.* (2003), the authors review various multiple testing procedures, which provide control of global error rates like the false discovery rate. We will intro-

duce two of these procedures in Section 3.2. In his comment to Ge *et al.* (2003), John D. Storey questions the assumption of non-equal null distributions. In the light of error rate control Storey argues that certain multiple testing procedures with pooling are equivalent to others without pooling. For example, if a procedure is based on scores and a *global* cutoff is used for significance analysis, differences between individual null score distributions are neglected and the procedure is similar to a procedure based on pooled p-values. A few proposals do not follow the global-cutoff approach. For example, Pollard and van der Laan (2004) base their procedure on observed scores and apply *individual* cutoffs to each of them, which is similar to gene-wise p-values. In their rejoinder following Ge *et al.* (2003) the authors argue in favor of non-equal null distributions and resulting assumptions, which are needed to show that a procedure provides control of an error rate. Storey *et al.* (2005) suggest that control might still be possible using pooled p-values even if the scores do not follow the same null distribution.

As stated in the outline of this thesis we do not focus on control of error rates but on their estimation. Pooled p-values offer certain advantages. First, their computation needs fewer permutations. When pooling, p-values are based on B times the number of genes permutations. Gene-wise p-values are based on B permutations only, thus the number of permutations B has to be large to achieve the granularity of pooled p-values. Second, pooled p-values monotonically decrease with increasing scores. Thus they directly mirror the estimated effect sizes—a feature, which made them popular among microarray experimenters.

The assumption of equal null distributions might be acceptable for the t-test score in Equation (2.8), where the scores of each gene ideally follow a t-distribution with certain degrees of freedom. In applications we observed little differences between gene-wise and pooled p-values. Typically, both p-values agree well when using t-scores. The situation is different for the z-score (2.10) or the log ratio score (2.4). These scores are not—or not properly—scaled for different variances thus we do not expect equal null distributions. We will investigate the differences between pooled and gene-wise p-values in Section 4.4. Although the two concepts lead to different p-values the overall p-value distributions appear to be very similar. The p-value distribution forms the basis of the false discovery rate estimation procedures introduced in Section 3.5 and of course of our own procedure introduced in Chapter 5. For our purposes, we take advantage of the faster computation and higher granularity of pooled p-values.

Inclusion of known covariates. In Chapter 6 we will turn to an inherent problem of permutation tests on biological data: in a typical experiment we observe a variable of interest, probably one or more known covariates, and certainly many unknown confounders. We can adjust for the known variables by including them into a regression model or using them as block variables (Smyth, 2004). In permutation tests, we might permute within blocks or restrict the procedure to those permutations that are balanced for the known variable. Including a covariate reduces the set of possible permutations but does not affect the general test procedure.

While we can adjust for known confounders, we cannot adjust for unknown confounders. These hidden variables introduce signal into the test procedure: suppose one random permutation correlates well with a hidden variable, which induces differential expression in many genes. Thus the distribution of scores derived from the correlated permutation might have heavier tails than the null distribution. In Chapter 6 we introduce an efficient procedure to discover and exclude these correlated permutations from the permutation test: if the permutation induces many genes—thus accumulating random scores in the tails of the null distribution—it is possible to identify this as a confounding signal.

However, the problem of confounding signal cannot be solved alone by using gene-wise instead of pooled p-values. Still the respective permutation correlates with the hidden variable and will lead to high scores for many genes. Within each genes' individual null distribution the respective random score of the correlated permutation might be in the tail and we would be able to identify this as an accumulation of individual null distributions with respective scores in the tails. As we will introduce in Chapter 6, we identify a confounding permutation by transforming its random scores to p-values and checking for accumulation of small p-values. It does not matter whether these p-values were derived by pooling or in the gene-wise fashion. An accumulation of high random scores—or low p-values—will be prominent in any case. We will explore the permutation approach in more detail in Chapter 6. For now, z-scores and pooled p-values are the basic ingredients for the methods introduced in the following chapters.