# Statistical Methods for Monte-Carlo based Multiple Hypothesis Testing

A thesis presented for the degree of

Doctor of Philosophy of Imperial College London

and the

Diploma of Imperial College

by

**Georg Hahn**

Department of Mathematics

Imperial College London

London SW7 2AZ

APRIL 2015

# Copyright

# Abstract

Statistical hypothesis testing is a key technique to perform statistical inference. The main focus of this work is to investigate multiple testing under the assumption that the analytical p-values underlying the tests for all hypotheses are unknown. Instead, we assume that they can be approximated by drawing Monte Carlo samples under the null.

The first part of this thesis focuses on the computation of test results with a guarantee on their correctness, that is decisions on multiple hypotheses which are identical to the ones obtained with the unknown p-values. We present `MMCTest`, an algorithm to implement a multiple testing procedure which yields correct decisions on all hypotheses (up to a pre-specified error probability) based solely on Monte Carlo simulation. `MMCTest` offers novel ways to evaluate multiple hypotheses as it allows to obtain the (previously unknown) correct decision on hypotheses (for instance, genes) in real data studies (again up to an error probability pre-specified by the user).

The ideas behind `MMCTest` are generalised in a framework for Monte Carlo based multiple testing, demonstrating that existing methods giving no guarantees on their test results can be modified to yield certain theoretical guarantees on the correctness of their outputs.

The second part deals with multiple testing from a practical perspective. We assume that in practice, it might also be desired to sacrifice the additional computational effort needed to obtain guaranteed decisions and to invest it instead in the computation of a more accurate ad-hoc test result. This is attempted by `QuickMMCTest`, an algorithm which adaptively allocates more samples to hypotheses whose decisions are more prone to random fluctuations, thereby achieving an improved accuracy.

This work also derives the optimal allocation of a finite number of samples to finitely many hypotheses under a normal approximation, where the optimal allocation is understood as the one minimising the expected number of erroneously classified hypotheses (with respect to the classification based on the analytical p-values). An empirical comparison of the optimal allocation of samples to the one computed by `QuickMMCTest` indicates that the behaviour of `QuickMMCTest` might not be too far away from being optimal.

# Acknowledgements

First and foremost, I would like to express my sincere thanks to my supervisor Dr Axel Gandy for all his expertise, support, help and advice.

I would also like to thank my friends for making the past years in London and at Imperial College London so worthwile and enjoyable, for all the countless hours of (Salsa) dancing socials and classes, language practice sessions, musicals and shows in London, excursions and everything else which does not fit into this short enumeration.

Finally, I would like to express my thanks to the Engineering and Physical Sciences Research Council (EPSRC) as well as to the Roth and DAAD (Deutscher Akademischer Austauschdienst, German Academic Exchange Service) scholarships for providing financial support.

*To all the voices in my head. We made it.*

C. Minas

# Table of contents

# List of Tables

# List of Figures

# List of Algorithms

# 1 Introduction

## 1.1 Preamble

Statistical hypothesis testing is a key technique to perform statistical inference. This is achieved by testing a single hypothesis or multiple hypotheses on observed data with the aim to determine if an observed outcome is unlikely to have occurred by chance alone. In this case, the result is called statistically significant.

Traditionally, a null hypothesis representing a conservative belief (for instance, an established opinion) is tested against an alternative. A discovery is made by rejecting the null hypothesis and thus the established belief in favour of the alternative.

As decisions based on data are not perfect, two error critera are usually used as guidelines for testing the null hypothesis. The type I error is the error occurring if a true null hypothesis is rejected, thus leading to a false finding. The type II error is the error associated to not rejecting a false null hypothesis. Typically, these two errors are not equally weighted in that more emphasis is placed on avoiding to reject a true null hypothesis (conservative belief), thus leading to type I error control.

A hypothesis is tested using a statistical test. A test statistic connects the hypothesis of interest with the observed data used to test it. Deriving a statistical test which is meaningful for testing a particular hypothesis is not straightforward.

To carry out an actual test, the distribution of the test statistic (called $T$) under the null hypothesis is needed. Given the distribution of $T$, its domain can be partitioned into the two sets of values for which the null hypothesis is rejected (also called the *critical region*) and non-rejected, where the critical region is chosen to have a probability equal to the type I error probability. A test result can be obtained by determining which region the test statistic evaluated on the observed data falls into.

An alternative but equivalent concept for testing a null hypothesis is the one of a *p-value*. Assuming a statistical test rejects for large values of $T$, the p-value $p = \mathbb{P}(T \geq t)$ encodes the probability of observing a realisation at least as large ("as extreme") as the one of the observed data. Rejecting a null with a p-value below the type I error threshold is equivalent to rejecting it based on the critical region. In a frequentist approach (Fisher), p-values are given without a threshold. This allows to assign a measure of significance to a hypothesis of interest without having to test it immediately: although the p-value itself does not permit a probability statement on the significance of an hypothesis, it allows to postpone the decision problem of rejecting or non-rejecting the hypothesis and to determine this decision later for a chosen threshold. This thesis will focus on such a test decision problem.

In many applications, neither the distribution of the test statistic under the null hypothesis nor the p-value can be computed analytically. They are therefore approximated using Monte Carlo techniques such as bootstrap or permutation tests. We call these tests Monte Carlo tests. A permutation test, for instance, is typically used to determine if two groups $A$ and $B$ are different in some aspect of interest encoded in the test statistic. To this end, the test pools $A$ and $B$ and randomly partitions the pooled sample into two groups again having sizes equal to the ones of $A$ and $B$ (this is often referred to as relabelling the data). Evaluating the test statistic exhaustively on all permutations allows to obtain a distribution for the null hypothesis that group labels do not matter, which in turn can be used to calculate a p-value.

Monte Carlo tests such as permutation tests are a popular means to test hypotheses as they exist for any test statistic. Importantly, the distribution of the test statistic under the null does not need to be known and in fact, it is often unobtainable in practice as underlying models for natural phenomena are rarely known. Moreover, modern computer power allows to easily approximate distributions of tests to high precisions via Monte Carlo simulation. This makes Monte Carlo tests a preferred choice to evaluate hypotheses even if a theoretical distribution could in principle be derived. Also, permutation tests take into account the (unknown) dependence between the tests which would often be very challenging to incorporate analytically.

So far, only one hypothesis was considered at a time. In practice, several hypotheses are often tested together, for instance when testing several gene regions of interest in a genome study. Testing multiple hypotheses, however, poses new challenges. Suppose 1000 hypotheses are to be tested together. If all of them were tested independently at a constant threshold controlling the type I error, such as $\alpha = 0.05$, the probability of committing at least one type I error would be $1-(1-\alpha)^{1000}$. This probability is almost one, meaning that one or more decisions will almost certainly be wrong. To counteract this phenomenon and to keep the error level at a pre-specified level, it is necessary to correct for multiple comparisons.

Several error criteria are available to evaluate hypotheses while correcting for multiple comparisons, for instance the *familywise error rate* (the probability of making one or more type I errors) or the less conservative *false discovery rate* (the expected proportion of false discoveries among all discoveries).

Special procedures have been developed to control such error critera, called multiple testing procedures. Two popular examples are the Bonferroni (1936) correction (Section 2.3.2) controlling the familywise error rate or the Benjamini and Hochberg (1995) procedure (Section 2.3.1) controlling the false discovery rate.

## 1.2 Motivation and aim of the thesis

We would like to test $m$ hypotheses $H_{01}, \ldots, H_{0m}$ for statistical significance using a multiple testing procedure given by a mapping

$$h : [0,1]^m \times [0,1] \to \mathcal{P}(\{1, \ldots, m\}) \tag{1.1}$$

which takes a vector of $m$ p-values $p \in [0,1]^m$ and a threshold $\alpha \in [0,1]$ and returns the set of indices of hypotheses to be rejected, where $\mathcal{P}$ denotes the power set. This procedure could, for instance, be the Bonferroni (1936) correction, the Sidak (1967) correction or the procedures of Holm (1979), Hochberg (1988) or Benjamini and Hochberg (1995).

Standard procedures require knowledge of the p-values of all tests. We assume that the p-values $p^* = (p_1^*, \ldots, p_m^*)$ of the underlying tests cannot be computed explicitly. For simplicity, we first assume in Chapter 2 that the threshold $\alpha^*$ at which we would like to test is constant and known, although this assumption is later relaxed in Chapter 3 to thresholds which may depend on the unknown $p^*$ and thus may be unknown themselves.

Even though $p^*$ is unknown, we assume that we can approximate $p^*$ and $\alpha^*$ (in the case where $\alpha^*$ is not given) through Monte Carlo simulations, for instance by resampling the data in case of bootstrap tests or by generating permutations when using permutation tests. This is the main assumption underlying all chapters of the thesis. The aforementioned scenario occurs widely in practical situations (Chen et al., 2013; Nusinow et al., 2012; Rahmatallah et al., 2012).

The aim of this thesis is to compute $h(p^*, \alpha^*)$ using Monte Carlo simulations only. Although the main focus lies on an exact computation, Chapters 4 and 5 also consider ways to approximate $h(p^*, \alpha^*)$ with high accuracy.

The motivation for trying to achieve the same classification as the one obtained with the p-values is mainly repeatability and objectivity of the results, which Gleser (1996) called *first law of applied statistics*: “Two individuals using the same statistical method on the same data should arrive at the same conclusion.” The statement of Gleser (1996) refers to the fact that Monte Carlo algorithms introduce additional randomness into an experiment which is not existent in the data. Moreover, the collection of Monte Carlo samples used by any such algorithm can introduce a substantial amount of extraneous error as a whole even though single Monte Carlo approximations might be reasonably precise. To minimise this influence and to not perturb the result, any additional randomness introduced by a Monte Carlo method has to be negligible. The following chapters will show ways to achieve control of the Monte Carlo variability in multiple hypothesis testing up to a guaranteed pre-specified error probability. Another reason for comparing to the p-values is that all the theoretical results of the multiple testing procedure based on the p-values still hold (again up to the guaranteed error probability).

It can additionally be remarked that setting the testing threshold also constitutes an arbitrary choice which influences the result. Although there is no gold standard for setting the threshold recent publications suggested revised guidelines to ensure meaningful significance levels (Johnson, 2013).

The importance of being able to compute $h(p^*, \alpha^*)$ with pre-specified precision is briefly demonstrated using an example which is treated in detail in Section 2.5.

We consider the classification of 9335 genes using real gene expression data from yeast chemostat cultivations (Knijnenburg et al., 2009). For each gene, this dataset contains 170 microarrays of yeast cultivations, divided into two classes: aerobically grown yeast and anaerobically grown yeast. Each gene becomes one hypothesis. The dataset is evaluated using the SAM (Significance Analysis of Microarrays) test statistic of Tusher et al. (2001) to analyse for which hypotheses the difference in expression between aerobically and anaerobically grown yeast is significant. As p-values are not directly available for this test statistic, a permutation test is used to approximate p-values.

We first draw a constant number of 1000 samples for each of the 9335 hypotheses, approximate all p-values as the proportion of exceedances and classify the dataset based on these p-value estimates. Repeating this 200 times shows that the decisions (rejections/ non-rejections) on most genes are fairly consistent. However, the decision on the particular gene *YLR139C*, previously reported as being significant by Rouillard et al. (1996) in the *Saccharomyces Genome Database* (Cherry et al., 2011), switches from significant to non-significant when repeatedly classified (each with roughly probability 0.5). Should *YLR139C* be claimed significant or insignificant?

This phenomenon occurs with many more hypotheses in real data studies and, more importantly, is not immediately resolved by drawing more samples as merely using more samples in conventional methods, albeit giving more accurate results, does not provide a guarantee on the correctness of each finding. The methods developed in Chapter 2 will classify gene *YLR139C* as being significant and provide a guarantee on the correctness of this result of at least 0.99 (chosen in advance by the user). To our knowledge, no other method available in the literature to date is able to resolve the correct classification of hypotheses (genes) in such a situation.

## 1.3 Brief literature review

Several algorithms published in the literature aim to approximate $h(p^*, \alpha^*)$ through Monte Carlo simulation for various multiple testing procedures $h$. Some of them aim to give guarantees on their result, but these guarantees are usually much weaker than the ones presented in this thesis.

A widely used naive method to approximate $h(p^*, \alpha^*)$ was already used in Section 1.2: the naive approach draws a constant number of $s$ samples for each of the $m$ hypotheses, approximates all p-values as the proportion of exceedances and classifies the dataset based on these p-value estimates. Though being simple, no guarantees on the correctness of the resulting decisions on individual hypotheses are usually given.

Guo and Peddada (2008) aim to improve upon the naive method by introducing an early stopping rule used to stop drawing the pre-specified total number of $s$ samples for certain hypotheses. This is achieved using bounds on the p-values of all hypotheses and a monotonicity property (Tamhane and Liu, 2008) of common multiple testing procedures. The authors guarantee that their test result, even with some hypotheses stopped from receiving all $s$ samples, is identical to the one of the naive method.

Though being related, the focus of the work of Guo and Peddada (2008) differs from the one of this thesis in that the authors do not provide any guarantee on how their test result relates to the one obtained with the underlying p-values $p^*$.

Jiang and Salzman (2012) present an early stopping procedure with a bound on its computational savings. As Guo and Peddada (2008), Jiang and Salzman (2012) aim at designing a procedure which gives the same result as the naive approach with a fixed number of samples. Moreover, the authors show that their procedure only controls the False Discovery Rate (FDR) (Benjamini and Hochberg, 1995) up to an error term.

The ad-hoc method of van Wieringen et al. (2008) stops generating samples for hypotheses for which a lower confidence level exceeds a pre-specified threshold (leading to a non-rejection). No early stopping for rejections is proposed. Being an ad-hoc method for a specific application, no explicit theoretical results are given.

The algorithm `MCFDR` of Sandve et al. (2011) is a modification of the algorithm of Besag and Clifford (1991), which, for a single hypothesis, stops drawing further samples when a fixed number of exceedances has been observed. The main idea of `MCFDR` is to use the criterion of Besag and Clifford (1991) to obtain quick non-rejections and to stop the entire algorithm once all remaining hypotheses are rejected based on their current estimated p-values. Although `MCFDR` gives quick results, neither `MCFDR` nor Besag and Clifford (1991) give any guarantees on how their test results relate to the result obtained with the p-values $p^*$.

The method proposed by Knijnenburg et al. (2009) uses ordinary permutation p-values if sufficiently many exceedances can be observed; otherwise, the authors approximate the p-values using a fitted extreme value distribution. The aim is to efficiently compute an estimate of all p-values, without giving any theoretical guarantees.

Moreover, several specialised resampling-based testing procedures for various sampling methods and various statistics can be found in Westfall and Young (1993). All above methods do not try to take the (unknown) dependence between the test statistics into account. Using permutation methods this can be attempted (Meinshausen, 2006; Westfall and Troendle, 2008).

## 1.4 Overview of all chapters

The contents of this thesis is as follows.

Chapter 2 introduces `MMCTest`, a sequential algorithm which gives, with arbitrarily high probability, the same classification as a specific multiple testing procedure applied to p-values $p^*$ at a constant testing threshold $\alpha^*$. The method can be used with a class of multiple testing procedures which includes the Benjamini and Hochberg (1995) False Discovery Rate (FDR) procedure and the Bonferroni (1936) correction controlling the Familywise Error Rate. One of the key features of the algorithm is that it stops the sampling for all the hypotheses which can already be decided as being rejected or non-rejected. `MMCTest` can be interrupted at any stage and then returns three sets of hypotheses: the rejected, the non-rejected and the undecided hypotheses. A simulation study motivated by actual biological data shows that `MMCTest` is usable in practice and that, despite the additional guarantee, it can be computationally more efficient than other methods.

The `MMCTest` algorithm was published in Gandy and H. (2014). Apart from the published contents, Chapter 2 contains the following additional sections: Section 2.5

presents an additional real data example showing how the ability of `MMCTest` to compute correct classifications up to a pre-specified error can be used in a genome study to reveal the (previously unknown) true decision on certain genes. Section 2.6 deals with the runtime of `MMCTest`. It shows that an algorithm computing complete classifications of all hypotheses using the Benjamini and Hochberg (1995) procedure with both a bounded error of misclassifications as well as a finite expected runtime cannot exist. Moreover, Section 2.6 proves that in connection with the Bonferroni (1936) correction, the runtime of `MMCTest` is finite when classifying all but two hypotheses.

The idea behind the `MMCTest` algorithm is generalised in Chapter 3 by providing a framework to test multiple hypotheses based on Monte Carlo simulation. This framework consists of a generic algorithm which incorporates several methods published in the literature, in particular all the methods aforementioned. We establish conditions which guarantee that the rejections and non-rejections obtained through Monte Carlo simulations are identical to the ones obtained with the p-values. Our framework is applicable to a general class of step-up and step-down procedures. Moreover, Chapter 3 shows how to use the framework to improve established methods in such a way as to yield theoretical guarantees on their results. These modifications can easily be implemented in practice and lead to a particular way of reporting multiple testing results as three sets together with an error bound on the correctness of all reported rejections and non-rejections, demonstrated exemplarily using a real biological dataset.

Chapter 4 looks at multiple testing from a practical perspective. In practice, it might be desirable to sacrifice the computational effort needed to obtain a guarantee on the test result and to invest it in a more precise ad-hoc classification instead – thereby trying to achieve as few misclassifications (erroneously classified hypotheses) as possible at the expense of having no guarantee on the correctness of the test result. The simple `QuickMMCTest` algorithm presented in Chapter 4 attempts this. It is based on Thompson (1933) Sampling and designed to adaptively allocate new samples to hypotheses whose p-values are closer to the testing threshold and thus whose decision is more prone to random fluctuations. The algorithm works with arbitrary step-up or step-down multiple testing procedures applied at a constant as well as a variable testing threshold. Such variable thresholds, for instance, might take account of the number of true null hypotheses. A simulation study demonstrates the higher accuracy of our approach in comparison to a variety of methods published in the literature, measured in numbers of erroneously classified hypotheses.

Finally, Chapter 5 revisits the `QuickMMCTest` algorithm designed to efficiently make use of a finite number of samples with the aim to minimise numbers of erroneously classified hypotheses. We are now interested in allocating a pre-specified total number of samples to all hypotheses in an optimal way – in the sense that the allocation minimises the total expected number of erroneously classified hypotheses. Neither using a constant number of samples per p-value estimate nor more sophisticated approaches available in the literature guarantee the computation of an optimal allocation in the above sense. Using the Kuhn-Tucker formalism, Chapter 5 derives the optimal allocation of a finite total number of samples to a finite number of hypotheses tested using the Bonferroni (1936) correction. Simulation studies indicate that the `QuickMMCTest` algorithm might not be too far away from asympotically imitating this optimal allocation.

The thesis ends with a conclusion in Chapter 6. Future work is discussed in Chapter 7. Additional material for each chapter such as further simulations and proofs can be found in the appendix where indicated.

## 1.5 List of publications

Chapter 2 of the thesis is published and the remaining chapters are available as articles on the *arXiv* preprint server:

- Chapter 2: Gandy and H. (2014), published in the *Scandinavian Journal of Statistics* and available on *arXiv:1209.3963*
- Chapter 3: Gandy and H. (2015a) on *arXiv:1402.3019*
- Chapter 4: Gandy and H. (2015c) on *arXiv:1402.2706*
- Chapter 5: Gandy and H. (2015b) on *arXiv:1502.07864*

# 2 MMCTest – A Safe Algorithm for Implementing Multiple Monte Carlo Tests

## 2.1 Introduction

This chapter introduces `MMCTest`, an algorithm to implement the multiplicity correction $h$ for multiple Monte Carlo tests. Using Monte Carlo samples only, the algorithm gives, with a pre-specified probability, the same classification (rejected and non-rejected hypotheses) as the classification based on the p-values $p^*$. For permutation tests, the p-values can in principle be obtained by running through all permutations. For bootstrap tests, the p-value is the probability that a bootstrapped test statistic is at least as extreme as the observed test statistic.

Our proposed algorithm is sequential: it starts with all hypotheses being unclassified and then takes samples and classifies hypotheses until all but a certain number of hypotheses have been classified or until a certain effort is reached. The proposed algorithm can be stopped earlier while having the same guarantee on the probability of misclassifications. When stopped before all hypotheses have been classified, the algorithm returns three sets: the rejected, the non-rejected and the not yet classified hypotheses.

Superficially, our algorithm is close to the algorithm proposed by Guo and Peddada (2008). Both algorithms maintain confidence intervals for the p-value of each hypothesis and stop generating samples for hypotheses for which a decision can be reached. For this both algorithms rely on the monotonicity property of the Benjamini and Hochberg (1995) procedure (Tamhane and Liu, 2008).

However, there are crucial differences. These mainly come from the different aim of the algorithms: Guo and Peddada (2008) aim to reduce the effort compared to the naive approach with a fixed number of samples per hypothesis. We aim to give the same classification as the classification using the p-values. As a consequence, their algorithm imposes an upper bound on the number of samples generated per hypothesis, whereas our algorithm is open-ended. Their algorithm does not aim to ensure repeatability, whereas we aim to do so. To be able to do this we judiciously control the joint coverage probability of the intervals.

To be specific, the main results in Guo and Peddada (2008, Proposition 1, Theorem 1) are related to Lemma 2.3 in the present chapter, with the difference that Guo and Peddada (2008) compare the classification to the naive approach with a fixed number of samples, whereas we compare the classification to the one based on the p-values. Furthermore, the chapter of Guo and Peddada (2008) has no equivalence to our Theorem 2.6 in which we prove that the classification returned by our algorithm converges to the one based on the p-values.

The basic `MMCTest` algorithm is described in Section 2.2. Moreover, Section 2.2

states conditions which bound the probability of classification errors and which guarantee the convergence of the testing result of `MMCTest` to the classification based on the p-values. The multiple testing procedure of Benjamini and Hochberg (1995) and the Bonferroni (1936) correction satisfy these conditions. This is shown in Section 2.3.

In Section 2.4, we first present an application of `MMCTest` motivated by real biological data, given by a microarray dataset of gene expressions for yeast chemostat cultivations (Knijnenburg et al., 2009) (Sections 2.4.1 to 2.4.2). Afterwards, we conduct simulation studies motivated by this real data in Sections 2.4.3 and 2.4.4 with the aim to compare the performance of a naive approach and of `MCFDR` (see Section 1.3) to `MMCTest`. Furthermore, we investigate the dependence of `MMCTest` on certain parameters (Sections 2.4.6 and 2.4.7).

The present chapter contains additional sections which are not included in its published version (Gandy and H., 2014). Section 2.5 revisits the application of `MMCTest` to the Knijnenburg et al. (2009) dataset and shows how `MMCTest` can be used to reveal the correct and previously unknown classification of certain genes (up to the pre-specified error probability). To our knowledge, no other method to date is able to resolve the correct classification of hypotheses in such a situation.

Section 2.6 looks at two aspects of the runtime of `MMCTest`. We prove that an algorithm computing a complete classification using the Benjamini and Hochberg (1995) procedure and with both a bounded error of misclassifications as well as a finite expected runtime cannot exist (Sections 2.6.1 and 2.6.2). Moreover, Section 2.6.3 proves that in connection with the Bonferroni (1936) correction, the expected runtime of `MMCTest` is finite when classifying all but two hypotheses.

We conclude with a discussion in Section 2.7. The `MMCTest` algorithm is implemented in an R-package (`simctest`, available on CRAN, The Comprehensive R Archive Network).

Most lemmas and theorems stated in this chapter are generalised in Chapter 3. In order to not state the proofs twice, most proofs are postponed to the more general versions in the next chapter.

Appendix A includes details on the simulation studies and an evaluation of a second dataset.

## 2.2 Description of the algorithm

### 2.2.1 Basic algorithm

Consider testing $m$ null hypotheses $H_{01}, \dots, H_{0m}$ having corresponding test statistics $T_1, \dots, T_m$ and observed values $t_1, \dots, t_m$. A large value of $t_i$ shall indicate evidence against $H_{0i}$.

We assume that for every hypothesis $H_{0i}$, where $i \in \{1, \dots, m\}$, we can obtain independent samples from the test statistic $T_i$ under the null hypothesis. We will denote these by $T_{ij}$, and the corresponding exceedance indicators will be denoted by $X_{ij} = \mathbf{1}(T_{ij} \geq t_i)$, $j \in \mathbb{N}$, where $\mathbf{1}$ is the indicator function. In the case of a permutation test, computing $T_{ij}$ involves generating permutations without replacement.

Testing is carried out using a multiple testing procedure $h$ defined in (1.1) at a constant and known threshold $\alpha^*$. For simplicity, we drop the dependence of $h$ on its second argument $\alpha$ in the entire chapter.

Following Tamhane and Liu (2008), we call a multiple testing procedure $h$ *mono-*

*tonic* if $h(p) \supseteq h(q)$ $\forall p \leq q$, where $p, q \in [0,1]^m$, i.e. if lower p-values lead to more rejections.

The following generic algorithm is designed for monotonic multiple testing procedures. It iteratively controls the set of hypotheses for which further samples need to be drawn by refining confidence intervals for every $p_i^*$ through Monte Carlo sampling. At iteration $n$, the confidence interval for the p-value $p_i^*$ is denoted by $I_i^n$. The upper confidence limit of a confidence interval $I_i^n$ is denoted by $\max I_i^n$ and the lower confidence limit is denoted by $\min I_i^n$.

The following variables and functions control the behaviour of the algorithm. The variable $\Delta$ controls how many additional samples are drawn in each iteration. It is increased geometrically by a constant $a \geq 1$ in each step of the algorithm, starting at $\Delta_0 \geq 1$. In the examples of this chapter we use $a = 1.25$ and $\Delta_0 = 10$. Two vectors $S, k \in \mathbb{N}_0^m$ keep track of counts.

The function $f(S, k, \Delta)$ computes a confidence interval for the p-value of a hypothesis based on the number of exceedances $S$ and the number of samples $k$ drawn for this hypothesis. The dependence on the current value of $\Delta$ is needed to be able to guarantee a joint coverage probability of all confidence intervals produced in the algorithm. For simplicity, we will assume that $f$ returns closed confidence intervals. In Section A.1 we give an example for such an $f$ which computes Clopper and Pearson (1934) confidence intervals and uses a spending sequence to guarantee an overall coverage probability.

The algorithm runs until at most $c \geq 0$ hypotheses remain unclassified or until the total number of samples drawn reaches a pre-specified limit $k_{\max}$. The following pseudo-code uses $c = 0$ and $k_{\max} = \infty$, thereby computing a classification of all hypotheses.

In the remainder of this chapter, $|\cdot|$ denotes the number of elements in a finite set and the length of an interval. Moreover, $\|\cdot\|$ denotes the Euclidean norm of a vector.

**Algorithm 2.1:** `MMCTest`

```
input: $(X_{ij})_{ij}, f, c = 0, k_{\max} = \infty, \Delta_0 = 10, a = 1.25$
$n \leftarrow 0$; $\Delta \leftarrow \Delta_0$; $\underline{A}_0 \leftarrow \emptyset$; $\overline{A}_0 \leftarrow \{1, \dots, m\}$;
$I_i^0 \leftarrow [0, 1]$; $S_i \leftarrow 0$, $k_i \leftarrow 0$ $\forall i = 1, \dots, m$;
while $|\overline{A}_n \setminus \underline{A}_n| > c$ and $\sum_{i=1}^m k_i \le k_{\max}$ do
    $n \leftarrow n + 1$;
    $\Delta \leftarrow \lfloor a\Delta \rfloor$;
    for $i \in \overline{A}_{n-1} \setminus \underline{A}_{n-1}$ do
        $S_i \leftarrow S_i + \sum_{j=k_i+1}^{k_i+\Delta} X_{ij}$;
        $k_i \leftarrow k_i + \Delta$;
        $I_i^n \leftarrow f(S_i, k_i, \Delta) \cap I_i^{n-1}$;
    for $i \notin \overline{A}_{n-1} \setminus \underline{A}_{n-1}$ do
        $I_i^n \leftarrow I_i^{n-1}$;
    $\underline{A}_n \leftarrow h((\max I_i^n)_{i=1,\dots,m})$;
    $\overline{A}_n \leftarrow h((\min I_i^n)_{i=1,\dots,m})$;
return $(\underline{A}_n, \overline{A}_n)$;
```

The algorithm works as follows: The number of additional samples $\Delta$ drawn in every step is increased geometrically. The total number of samples drawn up to iteration $n$ for a hypothesis $i \in \{1, \dots, m\}$ is stored in $k_i$ and the total number of observed exceedances is stored in $S_i$. For all hypotheses which are still under consideration, i.e. those in $\overline{A}_{n-1} \setminus \underline{A}_{n-1}$, an additional batch of $\Delta$ samples is drawn and new confidence intervals are computed. The confidence intervals remain unchanged for the other hypotheses. New classifications are then computed based on the updated upper and lower confidence limits.

The confidence intervals $I_i^n$ computed in Algorithm 2.1 are nested by construction.

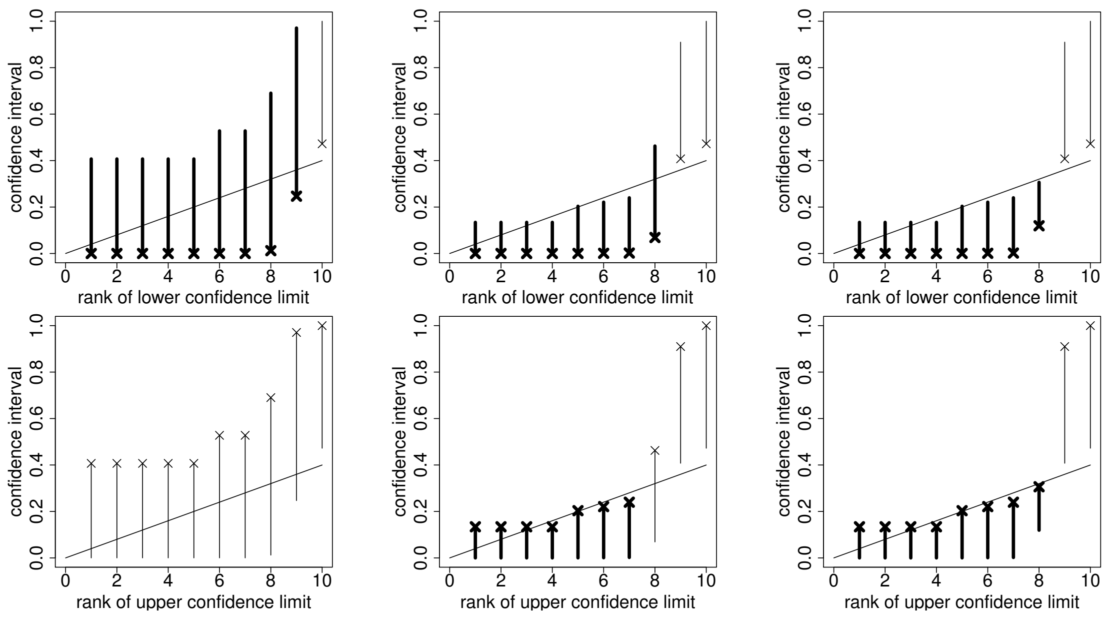


Figure 2.1: Example run of `MMCTest` on $m = 10$ hypotheses using the Benjamini-Hochberg procedure $h$: after the second iteration (left), after a few additional iterations (center) and after the last iteration (right). Bold confidence intervals denote elements of $\overline{A}_n$ in the upper row and elements of $\underline{A}_n$ in the lower row. The lower (upper) confidence limits used to compute $\overline{A}_n$ ($\underline{A}_n$) are marked with a cross.

**Example 2.2.** *An example run of `MMCTest` (with $m = 10$ hypotheses and $c = 0$) is shown in Figure 2.1. We use the Benjamini and Hochberg (1995) FDR controlling procedure (see Section 2.3.1) with threshold $\alpha = 0.4$ as the multiple testing function $h$. The function $f$ given in Section A.1 is used to compute confidence intervals. Columns show different iterations, the upper row shows the computation of $\overline{A}_n$, the lower row shows the computation of $\underline{A}_n$. The indices contained in $\overline{A}_n$ and $\underline{A}_n$ are visualised with bold confidence intervals. Additionally, the lower (upper) confidence limits used to compute $\overline{A}_n$ ($\underline{A}_n$) are marked with a cross. Only the lower (upper) end of the confidence interval matters for the computation of $\overline{A}_n$ ($\underline{A}_n$), thus the hypotheses are ordered by their lower (upper) confidence limit in the upper (lower) row.*

*In this example this turns out to be the same ordering. After the second iteration (left column),* `MMCTest` *has already classified the last hypothesis as being non-rejected as the lower confidence limit of its p-value lies above the line connecting the points* $(0,0)$ *and* $(m,\alpha)$ *which we call the Benjamini-Hochberg line (or threshold line). All other hypotheses are still undecided and thus their confidence intervals will be refined. After a few additional iterations (middle column), the seven smallest values can be classified as rejected as the upper confidence limit of the seventh value is below the line. Likewise, the confidence interval of the ninth value has now been shrunk to be entirely above the line which classifies this value as non-rejected. The eighth p-value is still unclassified as its confidence interval overlaps with the line. After refining the confidence interval further, the algorithm stops in the situation depicted in the right column with a complete classification* ($\overline{A}_n = \underline{A}_n$)*.*

The monotonicity of $h$ implies immediately that the sequence of sets $\underline{A}_n$ is increasing, that the sequence of sets $\overline{A}_n$ is decreasing and, on an additional assumption, that each $\underline{A}_n$ ($\overline{A}_n$) is a subset (superset) of the ideal set of rejections $h(p^*)$.

**Lemma 2.3.** *Assume that $h$ is monotonic.*

1. $(\underline{A}_n)_{n\in\mathbb{N}} \nearrow$ *and* $(\overline{A}_n)_{n\in\mathbb{N}} \searrow$.
2. *If* $p_i^* \in I_i^n\ \forall i,n$*, then* $\underline{A}_n \subseteq h(p^*) \subseteq \overline{A}_n\ \forall n \in \mathbb{N}$.

The statement of Lemma 2.3 also holds true for variable testing thresholds as proven in Lemma 3.4. Lemma 2.3 is thus a special case of Lemma 3.4 for a constant testing threshold.

### 2.2.2 Conditions and main results

In this section we show that under certain conditions the classification of `MMCTest` is correct with high probability, meaning that all classifications are identical to the classifications based on the p-values. Furthermore, we show that all hypotheses will be classified.

The first condition pertains to the multiple testing procedure $h$. Besides asking for monotonicity, it ensures that lowering the p-value of a rejected hypothesis or increasing the p-value of a non-rejected hypothesis does not change the result of $h$.

**Condition 2.4.** *1. $h$ is monotonic.*

*2. Let $p, q \in [0,1]^m$. If $q_i \leq p_i \; \forall i \in h(p)$ and $q_i \geq p_i \; \forall i \notin h(p)$, then $h(p) = h(q)$.*

The second condition requires the function $f$ to produce confidence intervals whose length goes uniformly to 0 as more samples are drawn.

**Condition 2.5.** *$|f(S,k,\Delta)|$ converges uniformly to 0 as $k \to \infty$, i.e. $\forall \epsilon > 0 \; \exists k_0 \in \mathbb{N}$ such that $\forall k \geq k_0$, $\forall S \in \{0, \dots, k\}$ and $\forall \Delta \in \mathbb{N}$, we have $|f(S,k,\Delta)| < \epsilon$.*

The main theorem guaranteeing convergence is as follows:

**Theorem 2.6.** *Suppose Conditions 2.4 and 2.5 hold and suppose that there exists $\delta > 0$ such that $p \in [0,1]^m$ and $\|p - p^*\| < \delta$ imply $h(p) = h(p^*)$. Then, on the event $\{p_i^* \in I_i^n \; \forall i, n\}$, both sequences $(\underline{A}_n)_{n \in \mathbb{N}}$ and $(\overline{A}_n)_{n \in \mathbb{N}}$ converge to $h(p^*)$, i.e. there exists $n_0 \in \mathbb{N}$ such that $\underline{A}_n = \overline{A}_n = h(p^*) \; \forall n \geq n_0$.*

Theorem 2.6 will be generalised in Theorem 3.6 to variable testing thresholds. The condition on $p^*$ in Theorem 2.6 ensures that $p^*$ has a neighbourhood on which $h$ is constant.

As shown in Section 2.3, the FDR controlling procedure of Benjamini and Hochberg (1995) and the Bonferroni (1936) correction both satisfy Condition 2.4. This is proven in Corollary 2.10 and Corollary 2.13. Moreover, they both satisfy the condition on $p^*$ in Theorem 2.6 (see Lemma 2.11 and Lemma 2.14) for almost all $p^*$.

The following third condition ensures that the confidence intervals computed by the function $f$ in Algorithm 2.1 have a guaranteed joint coverage probability. The choice of $f$ given in Section A.1 uses Clopper and Pearson (1934) confidence intervals and satisfies Condition 2.5 and Condition 2.7 (see Lemma A.1).

**Condition 2.7.** *For a given $\epsilon > 0$, the function $f$ computes confidence intervals $I_i^n$ in such a way that $\mathbb{P}(p_i^* \in I_i^n \ \forall i, n) \geq 1 - \epsilon$.*

The main theorem and Condition 2.7 together immediately give a bound on the probability of misclassifications.

**Corollary 2.8.** *Under the conditions of Theorem 2.6 and under Condition 2.7,*

$$\mathbb{P}(\exists n_0 : \underline{A}_n = h(p^*) = \overline{A}_n \ \forall n \geq n_0) \geq 1 - \epsilon,$$

*i.e. the probability that all classifications are correct is at least $1 - \epsilon$.*

*Proof.* By Theorem 2.6 we have $\underline{A}_n \to h(p^*)$, $\overline{A}_n \to h(p^*)$ as $n \to \infty$ conditional on $\{p_i^* \in I_i^n \ \forall i, n\}$. Under Condition 2.7, this event occurs with probability $\mathbb{P}(p_i^* \in I_i^n \ \forall i, n) \geq 1 - \epsilon$, hence $\mathbb{P}(\underline{A}_n \to h(p^*), \overline{A}_n \to h(p^*)) \geq 1 - \epsilon$. □

## 2.3 Some properties of multiple testing procedures

We discuss how and under which circumstances two multiple testing procedures, namely the Benjamini and Hochberg (1995) procedure and the Bonferroni (1936) correction, satisfy the conditions of Theorem 2.6.

In this section and the following sections, $A^c$ denotes the complement of $A \subseteq \{1, \dots, m\}$ with respect to $\{1, \dots, m\}$, where $m$ is the number of hypotheses under consideration.

### 2.3.1 Properties of the Benjamini-Hochberg procedure

The False Discovery Rate controlling procedure of Benjamini and Hochberg (1995) with threshold $\alpha > 0$ is defined as follows. Given $m$ p-values $p_1, \dots, p_m$, their order statistic is denoted by $p_{(1)} \leq p_{(2)} \leq \dots \leq p_{(m)}$. In case of a tie, equal values are assigned a rank in arbitrary order. Let $k$ be the largest index $i$ for which $p_{(i)} \leq \frac{i}{m}\alpha$. Then, rejecting all the hypotheses corresponding to $p_{(1)}, \dots, p_{(k)}$ ensures that the FDR is at most $\alpha$. The procedure can be expressed as

$$h(p) = \left\{ i \in \{1, \dots, m\} : \exists j : r_p(j) \geq r_p(i) \text{ and } m \frac{p_j}{r_p(j)} \leq \alpha \right\},$$

where $r_p(i)$ denotes the rank of $p_i$ in $p_{(1)} \leq p_{(2)} \leq \dots \leq p_{(m)}$.

The following lemma states three properties of the Benjamini-Hochberg procedure $h$ which are slightly stronger than Condition 2.4.

**Lemma 2.9.** *1. $h$ is monotonic.*

*2. Let $p, q \in [0,1]^m$. If $q_i \leq \frac{|h(p)|\alpha}{m}$ $\forall i \in h(p)$ and $q_i = p_i$ $\forall i \notin h(p)$, then $h(p) = h(q)$.*

*3. Let $p, q \in [0,1]^m$. If $q_i = p_i$ $\forall i \in h(p)$ and $q_i > \frac{\alpha}{m} r_p(i)$ $\forall i \notin h(p)$, then $h(p) = h(q)$.*

The statements of Lemma 2.9 actually apply to any step-up and, in a slightly modified fashion, to any step-down procedure. This is shown in Lemma B.1.

The second statement of Lemma 2.9 shows that all the p-values in the set of rejections can be increased up to a certain bound without affecting the result of $h$. The third statement states that $h$ stays invariant if the p-values in the non-rejection area are replaced by arbitrary values above the Benjamini-Hochberg line (see Example 2.2).

**Corollary 2.10.** *$h$ satisfies Condition 2.4.*

*Proof.* Statement 1 of Condition 2.4 is satisfied as $h$ is monotonic by Lemma 2.9.

To prove that the Benjamini-Hochberg procedure $h$ also satisfies the second statement of Condition 2.4, it suffices to show that for $p, q \in [0,1]^m$, both $q_i \leq p_i \; \forall i \in h(p)$ and $q_i = p_i \; \forall i \notin h(p)$ as well as $q_i = p_i \; \forall i \in h(p)$ and $q_i \geq p_i \; \forall i \notin h(p)$ imply $h(p) = h(q)$.

Indeed, let $p, q \in [0,1]^m$ be such that $q_i \leq p_i \; \forall i \in h(p)$ and $q_i = p_i \; \forall i \notin h(p)$. We have $p_i \leq \frac{|h(p)|\alpha}{m} \; \forall i \in h(p)$ by definition of $h$, thus $q_i \leq p_i \leq \frac{|h(p)|\alpha}{m} \; \forall i \in h(p)$ and $h(p) = h(q)$ by statement 2 of Lemma 2.9.

Similarly, let $p, q \in [0,1]^m$ be such that $q_i = p_i \; \forall i \in h(p)$ and $q_i \geq p_i \; \forall i \notin h(p)$. Using that $p_i > \frac{\alpha}{m} r_p(i) \; \forall i \notin h(p)$ it immediately follows that $q_i \geq p_i > \frac{\alpha}{m} r_p(i)$ $\forall i \notin h(p)$ and thus $h(p) = h(q)$ by statement 3 of Lemma 2.9. □

The next lemma states that $h$ is locally constant for almost all arguments:

**Lemma 2.11.** *If $p^* \in [0,1]^m$ with $p^*_{(i)} \neq i\alpha/m$, $i \in \{1, \ldots, m\}$, then there exists $\delta > 0$ such that $p \in [0,1]^m$ and $\|p - p^*\| < \delta$ imply $h(p^*) = h(p)$.*

*Proof.* The function $h$ stays invariant if all p-values do not change their rank outside of a tie and if no p-value crosses the Benjamini-Hochberg threshold line.

As $h$ is invariant to permutations, we may assume $p_1^* \leq \dots \leq p_m^*$. Let $\delta := \min\left(\left\{\frac{p_i^* - p_{i-1}^*}{2} : i = 2, \dots, m \text{ with } p_{i-1}^* < p_i^*\right\} \cup \left\{|p_i^* - \frac{i\alpha}{m}| : i = 1, \dots, m\right\}\right)$.

Let $p \in [0,1]^m$ with $\|p - p^*\| < \delta$. Then $p_{i-1}^* < p_i^*$ implies $p_{i-1} < p_{i-1}^* + \delta \leq p_i^* - \delta < p_i$. Thus, by possibly permuting indices corresponding to tied values in $p$, we may assume $p_1^* \leq \dots \leq p_m^*$ and $p_1 \leq \dots \leq p_m$. The ranks of the p-values in $p^*$ and $p$ are therefore the same.

Furthermore, $|p_i - p_i^*| < \delta \leq |p_i^* - i\alpha/m|$ for all $i \in \{1, \dots, m\}$, implying that $p_i^*$ and $p_i$ lie on the same side of the Benjamini-Hochberg line. Hence, $h(p^*) = h(p)$. $\square$

Lemma 2.11 shows that the condition on $p^*$ in Theorem 2.6 is satisfied for all the p-values except for those lying exactly on the Benjamini-Hochberg line.

### 2.3.2 Properties of the Bonferroni correction

The Bonferroni (1936) correction controls the Familywise Error Rate, defined by FWER $:= \mathbb{P}(V \geq 1)$, where $V$ is the number of true hypotheses which have been rejected (false positives). The method tests all $m$ hypotheses $H_{01}, \dots, H_{0m}$ at threshold $\alpha/m$ to guarantee FWER $\leq \alpha$. The Bonferroni correction $h_B$ returning the set of rejected indices can be stated as

$$h_B(p) = \{i \in \{1, \dots, m\} : p_i \leq \alpha/m\}.$$

Similarly to Lemma 2.9, the following lemma states two key properties of $h_B$ which are slightly stronger than the corresponding statements of Condition 2.4.

**Lemma 2.12.** *1. $h_B$ is monotonic.*

*2. Let $p, q \in [0,1]^m$. If $q_i \leq \frac{\alpha}{m}$ $\forall i \in h_B(p)$ and $q_i > \frac{\alpha}{m}$ $\forall i \notin h_B(p)$, then $h_B(p) = h_B(q)$.*

Similarly to Lemma 2.9, Lemma 2.12 is a special case of Lemma B.1 for the procedure of Bonferroni (1936).

The second statement of Lemma 2.12 shows that the result of $h_B$ is not affected if p-values in the rejection (non-rejection) area are replaced by arbitrary values below (above) the constant testing threshold $\alpha/m$.

**Corollary 2.13.** *$h_B$ satisfies Condition 2.4.*

The proof of Corollary 2.13 is similar to the one of Corollary 2.10. Moreover, Section 3.4.2 will show that Condition 2.4 is actually satisfied for a whole class of step-up and step-down procedures including the one of Bonferroni (1936).

Similarly to Lemma 2.11, the Bonferroni correction is locally constant for almost all values:

**Lemma 2.14.** *For all $p^* \in ([0, \alpha/m) \cup (\alpha/m, 1])^m$ there exists $\delta > 0$ such that $p \in [0,1]^m$ and $\|p - p^*\| < \delta$ imply $h_B(p^*) = h_B(p)$.*

*Proof.* Let $\delta := \min_{i \in \{1,\dots,m\}} |p^*_i - \alpha/m|$. Let $p \in [0,1]^m$ with $\|p - p^*\| < \delta$. For all $i \in \{1, \dots, m\}$, this implies that $p_i$ and $p^*_i$ lie on the same side of the threshold $\alpha/m$. Therefore, $h_B(p^*) = h_B(p)$. □

Lemma 2.14 shows that the condition on $p^*$ in Theorem 2.6 is satisfied for all the p-values except for those lying exactly on the threshold $\alpha/m$.

## 2.4 Simulation studies

This section starts with an overview of all simulation parameters used in the simulation studies (Section 2.4.1).

We then demonstrate that `MMCTest` can be used to classify thousands of hypotheses commonly encountered in real data studies (Section 2.4.2).

Moreover, this section shows that when matching the effort, `MMCTest` computes classifications containing a number of unclassified hypotheses which is comparable to the number of misclassifications incurred by current approaches like the naive method or the `MCFDR` algorithm – even though `MMCTest` is able to guarantee the correctness of all its classified hypotheses while for the two other methods, misclassified hypotheses typically remain unidentified in the testing result (Section 2.4.3 and Section 2.4.4). An ad-hoc variant of `MMCTest` computing a complete classification yields less misclassifications and random classifications than the other methods, demonstrating that `MMCTest` is the superior method for practical applications.

Section 2.4.5 illustrates the behaviour of the two sets $\underline{A}_n$ and $\overline{A}_n$ in a single run of `MMCTest`. Section 2.4.6 studies the dependence of the computational effort of `MMCTest` on the number of hypotheses $m$.

We conclude by empirically assessing the runtime of `MMCTest` in Section 2.4.7, demonstrating that whilst a complete classification can be computationally very expensive, most hypotheses can be classified with a reasonable effort.

### 2.4.1 The set-up

The following parameters were used throughout Section 2.4. The batch size $\Delta$ in Algorithm 2.1 is increased by $a = 1.25$ in every iteration, starting with $\Delta_0 = 10$. Confidence intervals are computed using the function $f$ with Clopper and Pearson (1934) confidence intervals and parameters $\epsilon = 0.01$ and $r = 10000$ (see Section A.1). The Benjamini-Hochberg procedure (at threshold $\alpha = 0.1$) as defined in Section 2.3.1 always serves as multiple testing procedure.

We measure the effort of any algorithm in terms of $N$, the total number of samples drawn during a run.

We use a yeast chemostat cultivation dataset of Knijnenburg et al. (2009). This dataset consists of 170 microarrays of yeast cultivations. The first 80 microarrays correspond to yeast which was grown aerobically, the second 90 microarrays correspond to yeast which was grown anaerobically. Every microarray reacts to 9335 genes, thus giving rise to 9335 null hypotheses (no effect of the gene onto the response). We evaluate this dataset using the SAM (Significance Analysis of Microarrays) test statistic of Tusher et al. (2001) in connection with a permutation test to analyse for which hypotheses the difference in expression between aerobically and anaerobically grown yeast is significant (see Section A.2).

To speed up the computation of the simulation studies in this and the following sections as well as to have an underlying "truth" for the Knijnenburg et al. (2009) dataset, we estimated each of the $m = 9335$ p-values once by generating $10^6$ permutations per hypothesis as outlined in Sections A.3 and A.4. Such a number of permutations is far more than what would commonly be used in practice. We then define these approximated p-values to be the true underlying p-values $p_1^*, \dots, p_m^*$ we are interested in, although they do not necessarily have to be equal to the p-values underlying each hypothesis. A plot of the p-values $p_1^*, \dots, p_m^*$ is given in Section A.4.

In the following sections, we draw Bernoulli samples with success probabilities $p_1^*, \dots, p_m^*$ instead of generating actual permutations. The classification obtained by applying the Benjamini-Hochberg procedure directly to the p-values $p_1^*, \dots, p_m^*$ is used to compute misclassifications.

Section A.5 contains another comparison of `MMCTest` to the naive method and to `MCFDR` on a simulated dataset with a larger proportion of true null hypotheses than the one of the dataset of Knijnenburg et al. (2009), broadly confirming the results of Sections 2.4.3 and 2.4.4.

### 2.4.2 Application to Real Data

`MMCTest` is applied once to the p-values as described in Section 2.4.1.

After having drawn $24.5 \cdot 10^6$ samples all but 100 hypotheses are classified. This corresponds to only around 2600 samples per hypothesis, thus making a classification with such a precision fairly easy to compute. Drawing roughly the same number of samples again (a total number of $49.7 \cdot 10^6$ samples) classifies all but 50 hypotheses.

`MMCTest` can be stopped whenever the user's desired number of classifications is achieved. All but 20 hypotheses are classified after $159 \cdot 10^6$ samples and all but 10 hypotheses after $255 \cdot 10^6$ samples. A classification of all but 5 hypotheses is obtained after having drawn a total number of $12 \cdot 10^9$ samples. This is, of course, extremely computationally intensive. The total number of samples drawn for a classification of all but 5 hypotheses corresponds to roughly $1.3 \cdot 10^6$ samples per hypothesis.

Table 2.1: Comparison of the naive method to `MMCTest`

| naive method | | | | MMCTest | | |
|---|---|---|---|---|---|---|
| | | | | guaranteed classification | forced classification | |
| *s* | *mis* | *rc* | *N* | *unclassified hypotheses* | *mis* | *rc* |
| 100 | 238 | 680 | 933500 | 7677 | 236 | 686 |
| 1000 | 69 | 230 | 9335000 | 317 | 20 | 62 |
| 10000 | 21 | 65 | 93350000 | 32 | 3 | 6 |

*s*: number of samples used by the naive method for each hypothesis; *mis*: average number of misclassifications; *rc*: number of randomly classified hypotheses; *N*: average total number of samples.

A comparison to the classification result obtained by applying the Benjamini and Hochberg (1995) procedure to the p-values $p_1^*, \dots, p_m^*$ shows that in all the classifications previously reported, none of the decided hypotheses was wrongly classified.

### 2.4.3 Comparison to the naive method

We compare `MMCTest` to the sampling scheme which draws a constant number of samples $s$ for each hypothesis. It then estimates each p-value via its proportion of exceedances (a formula for this estimate is given in Section A.3) and computes a classification by applying the multiplicity correction to the estimates, thus treating the estimated p-values as if they were the p-values. We will call this the naive method. The naive method is widely used in connection with the False Discovery Rate approach to evaluate real biological data (Cohen et al., 2012; Gusenleitner et al., 2012; Nusinow et al., 2012; Rahmatallah et al., 2012).

The results presented in this and the following section are based on 10000 runs. In each run, we draw Bernoulli samples for the naive method and for `MMCTest` as described in Section 2.4.1. The sampling standard deviation of averages is less than the least significant digit we report in tables.

Table 2.1 shows the simulation results. The second column displays the average number of misclassifications for the naive method. A considerable number of misclassifications occurs; even when using $s = 10000$ samples to estimate each p-value about 21 misclassifications still occur on average for the naive method.

The third column in Table 2.1 shows an alternative criterion, the number of randomly classified hypotheses ($rc$), which we define as follows. Let $f_i^r$ be the proportion of a rejection of hypothesis $H_{0i}$ in the 10000 runs. A hypothesis $H_{0i}$ is considered to be randomly classified if $\min(f_i^r, 1 - f_i^r)$ is strictly larger than 0.1.

The number of randomly classified hypotheses is substantially larger than the average number of misclassifications. This demonstrates that for a substantial number of hypotheses, the decision reported is mainly determined by the Monte Carlo randomness (as opposed to the relation of $p^*$ to the threshold).

The total number of samples $N$ drawn during each run of the naive method is given in the fourth column of Table 2.1.

`MMCTest` is run on the p-values (see Section 2.4.1) using at most the total number of samples the naive method had used. The fifth column in Table 2.1 shows the average number of remaining unclassified hypotheses upon termination.

The average number of unclassified hypotheses of `MMCTest` is larger than the number of misclassifications of the naive method. However, `MMCTest` gives a result which is proven to be reliable with pre-specified probability in contrast to the one computed by the naive method. For large values of $s$, `MMCTest` yields average numbers of unclassified hypotheses which almost equal the number of misclassifications observed for the naive method even though `MMCTest` guarantees the correctness of its classified hypotheses while the misclassifications in the testing result of the naive method typically remain unidentified.

As shown in Table 2.1, at a high precision, `MMCTest` yields less unclassified hypotheses than the naive method yields randomly classified hypotheses. This indicates that `MMCTest` gets competitive for a realistic precision and starts overtaking the naive method for multiple testing settings which are evaluated at high precision.

`MMCTest` is stopped on reaching the number of samples used by the naive method. Nevertheless, all the theoretical guarantees stated in Section 2.2 are still valid, but not all hypotheses are being classified. A complete classification in an ad-hoc fashion can be obtained by applying the multiple testing procedure $h$ to the p-value estimates $\hat{p}_i = (S_i+1)/(k_i+1)$ after stopping ($S_i$ and $k_i$ are as in Algorithm 2.1). The theoretical guarantees of Section 2.2 are not valid any more for the ad-hoc procedure.

The two last columns of Table 2.1 show the average number of misclassifications and the number of randomly classified hypotheses for the ad-hoc procedure which forces a complete classification upon termination. With this simple modification, `MMCTest` yields considerably lower numbers of misclassifications and randomly classified hypotheses for a high precision than the naive method.

The forced classification should only be used if a complete classification is needed within a limited effort. In all other cases, whenever the algorithm is stopped, we recommend using the partioning of the hypotheses into rejected, non-rejected and not classified hypotheses as testing result of the algorithm.

### 2.4.4 Comparison to `MCFDR`

We now focus on a comparison of `MMCTest` to `MCFDR` of Sandve et al. (2011), given in Table 2.2. `MCFDR` is run first on the p-values (see Section 2.4.1) already used for

Table 2.2: Comparison of `MCFDR` to `MMCTest`

| MCFDR | | | | MMCTest | | |
|---|---|---|---|---|---|---|
| | | | | guaranteed classification | forced classification | |
| *u* | *mis* | *rc* | *N* | *unclassified hypotheses* | *mis* | *rc* |
| 10 | 172 | 524 | $1.3 \cdot 10^6$ | 7547 | 208 | 592 |
| 20 | 123 | 389 | $2.2 \cdot 10^6$ | 7268 | 130 | 398 |
| 50 | 80 | 267 | $5.3 \cdot 10^6$ | 763 | 50 | 168 |
| 100 | 57 | 186 | $10.5 \cdot 10^6$ | 288 | 18 | 54 |
| 200 | 40 | 128 | $21.0 \cdot 10^6$ | 132 | 9 | 29 |
| 500 | 25 | 72 | $52.5 \cdot 10^6$ | 49 | 4 | 8 |
| 1000 | 18 | 54 | $104.9 \cdot 10^6$ | 30 | 3 | 6 |

*u*: number of test statistics exceeding the reference statistic (tuning parameter of `MCFDR`); *mis*: average number of misclassifications; *rc*: number of randomly classified hypotheses; *N*: average total number of samples.

the comparison of `MMCTest` to the naive method and `MMCTest` is then applied with matched effort. The `MCFDR` algorithm has one tuning parameter: the number $u$ of test statistics exceeding the reference statistic before stopping (this number was called $h$ in Sandve et al. (2011)). In Sandve et al. (2011) the authors recommend using $u = 20$, but we will also consider larger values.

In its original statement in Sandve et al. (2011), the `MCFDR` algorithm uses a modification of the Benjamini-Hochberg procedure of Pounds and Cheng (2006) which uses an estimate $\hat{\pi}_0(p)$ of the proportion of true null hypotheses. `MCFDR` can also be used together with the standard Benjamini-Hochberg procedure by setting $\hat{\pi}_0(p)$ to one. The following results have been computed using the standard Benjamini-Hochberg procedure (as defined in Section 2.3.1) for both `MCFDR` and `MMCTest`.

The first columns of Table 2.2 show the average number of misclassifications *mis* and the number of randomly classified hypotheses *rc* for `MCFDR` for various values of $u$. Similar to Table 2.1, the number of randomly classified hypotheses occurring for `MCFDR` is generally larger than the average number of misclassifications.

When using `MMCTest`, the number of unclassified hypotheses is generally larger than the number of misclassifications for `MCFDR`. The advantage of using `MMCTest` is, as before, the guaranteed classification.

`MMCTest` becomes more competitive for higher precisions. For large values of $u$, the `MMCTest` algorithm classifies all hypotheses with confidence up to a number which almost equals the number of misclassifications of `MCFDR`, and which is less than the number of randomly classified hypotheses of `MCFDR`.

The forced classification in `MMCTest` (the last two columns in Table 2.2) yields a considerably better classification than `MCFDR` for high precisions, both in terms of misclassifications and randomly classified hypotheses.

### 2.4.5 Progression of classifications of `MMCTest`

Figure 2.2 illustrates the size of the sets $\underline{A}_n$ and $\overline{A}_n$ in a single run of `MMCTest` on the p-values. The parameters for `MMCTest` we used are the same as the ones given in Section 2.4.1. `MMCTest` is run until all but $c = 10$ hypotheses are classified.

For the p-values used as the underlying "truth" (see Section 2.4.1, a plot is available in Figure A.1) we expect a large proportion of hypotheses to be easily classified to lie above the Benjamini-Hochberg line with threshold $\alpha = 0.1$. Indeed, the size of $\overline{A}_n$ drops quickly during the first iterations.

As we use a relatively low threshold of $\alpha = 0.1$, a considerable effort is needed to shrink the confidence intervals of the rejected hypotheses in such a way as to make them lie entirely below the Benjamini-Hochberg line. This becomes visible in Figure 2.2 as the size of $\underline{A}_n$ remains unchanged over a large period of time and increases only at a later stage.

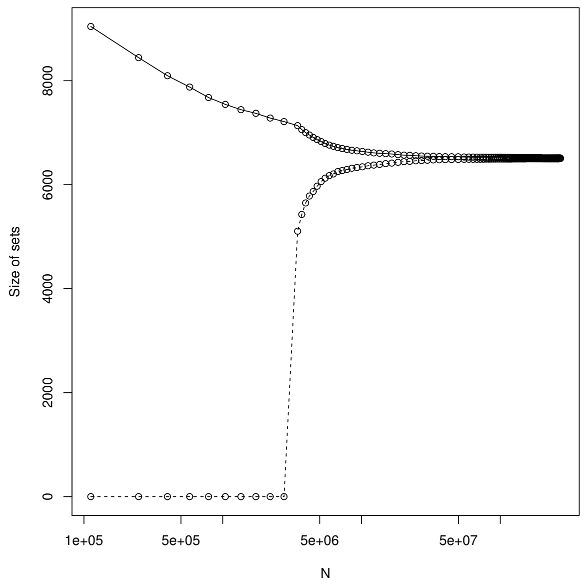


Figure 2.2: Threshold $\alpha = 0.1$. Size of $\underline{A}_n$ (dashed) and $\overline{A}_n$ (solid) as a function of the total number of samples $N$ drawn in a single run of `MMCTest`. The end of every iteration is indicated by a circle. The horizontal axis has a log-scale.

The sudden increase in size of set $\underline{A}_n$ in Figure 2.2 shows that several hypotheses are classified together.

### 2.4.6 Dependence of the effort on the number of hypotheses

How does the number of samples $N$ depend on the number of hypotheses?

Figure 2.3 shows 50%-, 95%- and 99%-quantiles of the effort $N$ for a classification of $m$ hypotheses, where $m$ ranges from 500 to 10000 in steps of 100. Quantiles are computed based on 10000 repetitions. For each value of $m$ and each repetition, a new p-value distribution is obtained by resampling with replacement from the the fixed p-values $p_1^*, \ldots, p_m^*$ (see Section 2.4.1). `MMCTest` is then run on the new distribution obtained in this way until all but $c = 0.01m$ hypotheses are classified.

Figure 2.3 indicates that the effort $N$ for a classification of all but $c = 0.01m$

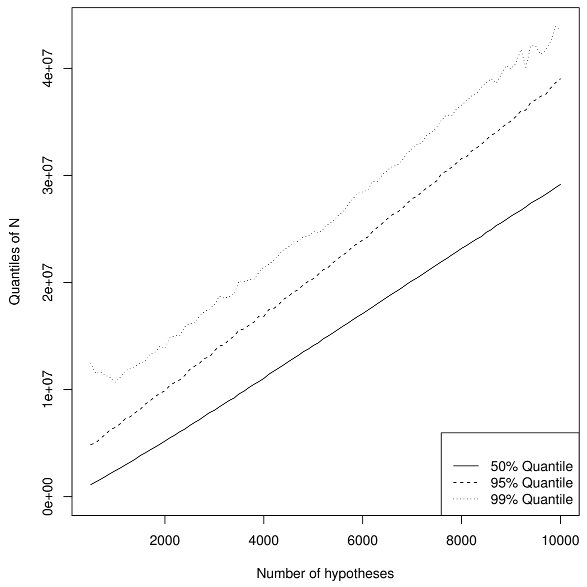


Figure 2.3: 50%-, 95%- and 99%-quantiles of the effort $N$ against the number of hypotheses $m$. Quantiles are computed based on 10000 runs classifying all but $c = 0.01m$ hypotheses. P-values for various values of $m$ are obtained by resampling with replacement from the fixed p-values $p_1^*, \dots, p_m^*$ (see Section 2.4.1).

hypotheses increases linearly in $m$.

### 2.4.7 Dependence of the effort on the number of unclassified hypotheses

Figure 2.4 shows the dependence of the effort $N$ on the number of unclassified hypotheses for a fixed number of hypotheses $m$. The right hand side of Figure 2.4 corresponds to the situation of all hypotheses being unclassified. The classification becomes more complete as the quantile curves approach the left hand side.

To generate this figure, `MMCTest` is applied 1000 times in the following way to the p-values $p_1^*, \dots, p_m^*$ ($m = 9335$, see Section 2.4.1): The current size $c$ of the set $\overline{A}_n \setminus \underline{A}_n$ and the current total number of samples $N_c$ are recorded in each iteration

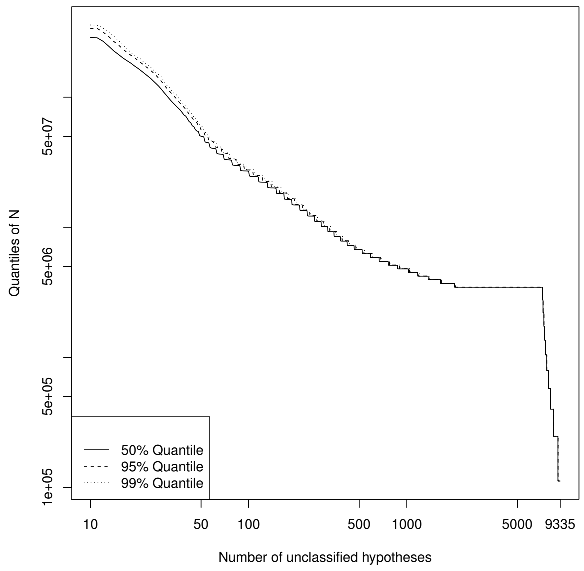


Figure 2.4: Effort $N$ against number of unclassified hypotheses. 50%-, 95%- and 99%-quantiles of $N$ based on 1000 simulations. Log-scale on both axes. The classification becomes more complete as the quantile curves approach the left hand side.

$n$. If several p-values are classified together in an iteration, some $c$ do not have a corresponding $N_c$. To be conservative, a missing value $N_c$ is set to $N_{c'}$ for the largest $c' < c$ for which $N_{c'}$ is not missing. Each time the algorithm is run until all but $c = 10$ hypotheses are classified.

The effort is reasonable for classifying all but a few hypotheses. Classifying the last few hypotheses seems to be computationally intensive.

The steps in Figure 2.4 are caused by several hypotheses with p-values far off the Benjamini-Hochberg line being classified together. This effect also occurs in Figure 2.2 which shows that at a certain iteration $n$, several hypotheses are classified together, thereby causing a sudden increase in the size of the set $\underline{A}_n$.

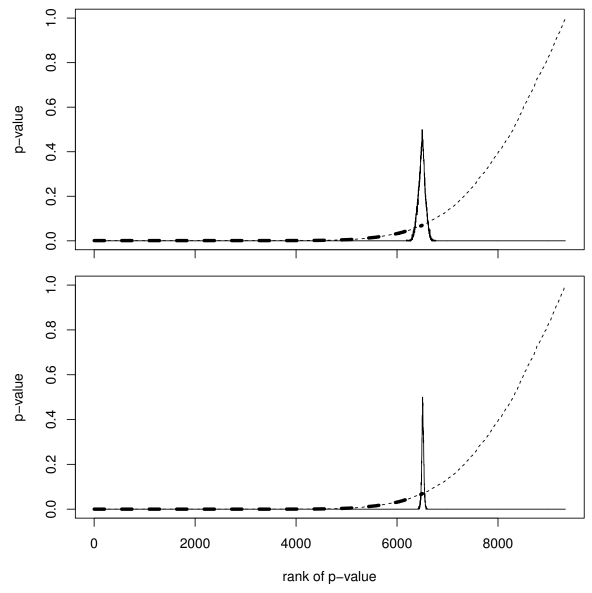


Figure 2.5: Estimated p-values belonging to significant (bold dashed) and non-significant (dashed) genes as well as probabilities of being randomly classified (solid curve). P-values are computed using 1000 (top) as well as 10000 (bottom) permutations and random classification probabilities are based on 400 repetitions. Multiple testing via Benjamini and Hochberg (1995) at threshold 0.1.

## 2.5 An application of multiple testing to gene expression data

The following example demonstrates the capability of `MMCTest` to reveal the correct classification of hypotheses (in this case of genes) with pre-specified probability – a feature not provided by other methods available in the literature to our knowledge.

We re-consider the classification of 9335 genes using real gene expression data from yeast chemostat cultivations (Knijnenburg et al., 2009) investigated in Section 2.4. As before, we use SAM (Significance Analysis of Microarrays) of Tusher et al. (2001) as the test statistic. As p-values are not directly available for this test statistic, a permutation test is used instead.

We are interested in the probability of a random classification, meaning the probability that a gene switches between being classified as significant or non-significant when classified several times. For this, we generated $N \in \{1000, 10000\}$ permutations per gene, approximated the p-value using these and then classified all genes by applying the Benjamini and Hochberg (1995) procedure at threshold 0.1. This was repeated $r = 400$ times. As before, we compute the empirical probabilities $f_i^r$ ($f_i^n = 1 - f_i^r$) that gene $i$ is rejected (non-rejected) in these $r$ repetitions and use $\min(f_i^r, f_i^n) \in [0, 0.5]$ as probability of being randomly classified.

Figure 2.5 shows the average estimated p-values over all $r$ runs using 1000 (top) and 10000 (bottom) permutations as well as significant (bold dashed) and non-significant (dashed) genes. The classification was obtained by applying the Benjamini and Hochberg (1995) procedure to these average p-values. For each gene, Figure 2.5 also displays the probability of being randomly classified (solid curve).

Figure 2.5 shows that most genes have a probability of being randomly classified of almost zero. They can therefore be unambiguously classified as being either significant or non-significant. However, a considerable number of genes with a p-value in the neighbourhood of the last significant and first non-significant gene have a probability of being randomly classified of up to 0.5, even when using $N = 10000$ permutations to estimate each p-value.

Looking up genes which are reported to be significant in the *Saccharomyces Genome Database* (Cherry et al., 2011) based on previous studies reveals that almost all genes which have been identified so far have a zero probability of being randomly classified.

However, individual probabilities based on estimated p-values with 1000 permutations show that gene *YJL209W*, previously reported to be significant in Chen and Dieckmann (1997), has a non-zero probability (0.01) of being randomly classified and gene *Q0250* (Fox, 1979) is randomly classified with probability 0.24. More importantly, gene *YLR139C*, previously reported to be significant in Rouillard et al. (1996), is randomly classified with probability 0.48.

Using p-value estimates with 10000 permutations shows that *YLR139C* (Rouillard et al., 1996) still remains to be randomly classified with probability 0.21. The two other genes mentioned before are not randomly classified with the increased number of permutations.

This phenomenon also depends on the testing threshold. When using the Benjamini and Hochberg (1995) procedure at the threshold 0.05 and 1000 permutations to estimate p-values, genes *Q0060* (DDB, 2001), *Q0105* (Kreike et al., 1979) and *Q0275* (Meunier, 2001) have high probabilities of being randomly classified (0.12, 0.12 and 0.25, respectively). When using 10000 permutations at the threshold 0.05, only gene *Q0275* (Meunier, 2001) remains randomly classified with non-zero probability (0.02).

The decision made on these genes therefore depends on the Monte Carlo error to a considerable degree and not on the actual data.

We used the implementation of `MMCTest` in the package `simctest` on CRAN with default values (in particular using a pre-specified error probability of $\epsilon = 0.01$) to resolve the classification of the genes *YJL209W*, *Q0250* and *YLR139C* by recomputing a complete classification of all hypotheses.

After the first iteration (for the default batch size of our *R* implementation this amounts to a total number of 112020 permutations), the genes *YJL209W* and *Q0250* were classified by `MMCTest` as being non-significant. The fact that both genes had a vanishing probability of being randomly classified when estimating p-values with 10000 permutations explains why they could be classified easily.

The gene *YLR139C* was much harder to classify. After having drawn a total number of $1.7 \cdot 10^8$ permutations over all hypotheses, it was classified as being significant by `MMCTest`. With probability at least 0.99 these decisions are correct in the sense that they are identical to the ones obtained if underlying p-values had been available.

## 2.6 Expected runtime of `MMCTest`

This section comments on the runtime of `MMCTest`. First, under suitable assumptions, an argument similar to the one used in Gandy (2009) establishes an infinite expected runtime for the classification of one hypothesis in Section 2.6.1.

Section 2.6.2 applies this result to multiple testing using the Benjamini and Hochberg (1995) procedure. It shows that an algorithm which classifies all but a fixed number $c \geq 0$ of hypotheses and which features both bounded error of misclassifications and finite expected runtime cannot exist.

Though a complete classification takes an infinite expected runtime, the expected runtime is finite for shrinking all but two confidence intervals to such a length as to not make them overlap with the threshold line. This is proven in Theorem 2.16 in Section 2.6.3. For the Bonferroni (1936) correction, `MMCTest` will thus terminate in finite expected runtime if all but two hypotheses need to be classified.

### 2.6.1 A classification of a random p-value takes infinite expected runtime

Under conditions, deciding one hypothesis having a p-value which is random takes an infinite expected runtime. The derivation included in this section is similar to the one in Gandy (2009).

Consider deciding $m = 1$ hypothesis $H_{01}$ with a random p-value $p_1$. Assume that $H_{01}$ is tested with a sequential algorithm approximating $p_1$ which gives a guarantee of $1 - \epsilon$ on the correctness of its decision as in Corollary 2.8.

Computing a decision on the single hypothesis $H_{01}$ is equivalent to deciding whether $p_1$ lies above or below a testing threshold $\alpha$. For some $p_1 > \alpha$, consider testing $H_0 : p = \alpha$ against $H_1 : p = p_1$. A test can be constructed by rejecting $H_0$ if and only if the algorithm used for classifying $H_{01}$ reports *not rejected*. By the above assumption (Corollary 2.8), both the Type 1 and the Type 2 error of this test are $\epsilon$.

For such a sequential test, a lower bound on the expectation of the number $N$ of steps is given in Wald (1945) by

$$\mathbb{E}(N|p = p_1) \geq \frac{\epsilon \log\left(\frac{\epsilon}{1-\epsilon}\right) + (1-\epsilon)\log\left(\frac{1-\epsilon}{\epsilon}\right)}{p_1 \log\left(\frac{p_1}{\alpha}\right) + (1-p_1)\log\left(\frac{1-p_1}{1-\alpha}\right)}.$$

Abbreviate the numerator by $C$ and consider $p_1$ as random in a Bayesian setup, having distribution function $F(p_1)$ with derivative $F'(\alpha) > 0$. Assume that for a suitable $\gamma > 0$, there exists a constant $d > 0$ such that $F'(p_1) \geq d$ in $(\alpha, \alpha+\gamma)$. Then the expected runtime is bounded by

$$\begin{aligned}\mathbb{E}(N) &= \int_0^1 \mathbb{E}(N|p = p_1)dF(p_1) \geq \int_\alpha^{\alpha+\gamma} \mathbb{E}(N|p = p_1)dF(p_1) \\ &\geq C \cdot d \cdot \int_\alpha^{\alpha+\gamma} \left(p \log\left(\frac{p}{\alpha}\right) + (1-p)\log\left(\frac{1-p}{1-\alpha}\right)\right)^{-1} dp = \infty,\end{aligned}$$

as the integrand is proportional to $(p-\alpha)^{-2}$ as $p \to \alpha$.

As the aforementioned assumptions on the algorithm used to derive the test on $H_0$ apply to `MMCTest`, its expected runtime for classifying one hypothesis with random p-value as above is infinite.

## 2.6.2 Extension to infinite expected runtime for multiple testing

The result of Section 2.6.1 can be extended to multiple hypothesis testing.

Assume that multiple testing is carried out using the Benjamini and Hochberg (1995) procedure and that p-values are randomly distributed. This section shows that under reasonable assumptions on the distribution of p-values, an algorithm which classifies all but a fixed number $c \geq 0$ of hypotheses and which features both bounded error of misclassifications and finite expected runtime cannot exist.

Consider testing $m \geq 1$ hypotheses $H_{01}, \ldots, H_{0m}$. Suppose that the p-values $p_1, \ldots, p_m$ for $H_{01}, \ldots, H_{0m}$ have a joint distribution with support $[0,1]^m$ and multivariate density $f_p(p)$. The density $f_p(p)$ shall have the property that for all $\delta > 0$, there exists a $\tau > 0$ such that $f_p(p) > \tau$ on $[\delta, 1-\delta]^m$. This condition is satisfied for many common densities.

For a given threshold $\alpha$ and $0 < \gamma < \alpha/(4m)$, define the set

$$A := \left[\frac{\alpha(m-1)}{m} + 2\gamma, \alpha - 2\gamma\right]^{m-1} \times [\alpha - \gamma, \alpha + \gamma].$$

Moreover, let $B := [\delta, 1-\delta]^m$ and choose $\delta$ in such a way that $A \cap B \neq \emptyset$.

Consider drawing a vector $\tilde{p} = (\tilde{p}_1, \ldots, \tilde{p}_m)$ of p-values from $f_p$ conditional on being in $A$. By definition of $A$, $\tilde{p}_m$ is the largest value in $\tilde{p}$. As $\tilde{p}_m$ is at most a distance $\gamma$ away from the threshold line and all other values are at least $2\gamma$ away, $\tilde{p}_m$ is also the closest p-value to the line. The value $\tilde{p}_m$ may lie above or below the line, all smaller values are surely above the line.

In this scenario, none of the hypotheses can be classified unless it is known whether $\tilde{p}_m$ lies above or below the line. As shown in Section 2.6.1, the runtime needed to decide $H_{0m}$ and thus all hypotheses is infinite. Thus $\mathbb{E}(N|\tilde{p} \in A) = \infty$, where $N$ denotes the number of samples.

By the Law of Total Expectation, the runtime can then be expressed as

$$\mathbb{E}(N) \geq \mathbb{E}(N|p \in A) \cdot \mathbb{P}(p \in A) = \infty,$$

where it was used that $\mathbb{E}(N|p \in A) = \infty$ and that $f_p(p) > \tau > 0$ on $B$, thus $\mathbb{P}(p \in A) \geq \mathbb{P}(p \in A \cap B) > 0$.

The above consideration shows that for multiple testing using the Benjamini and Hochberg (1995) procedure, an infinite expected runtime is needed even if not all hypotheses are required to be classified. Similar constructions can be made for other step-up and step-down procedures.

### 2.6.3 A finite expected runtime can be achieved for testing with the Bonferroni (1936) procedure

Although a finite expected runtime for Benjamini and Hochberg (1995) testing cannot be achieved, it can be proven that `MMCTest` decides all but two hypotheses in finite expected time for the Bonferroni (1936) procedure.

To be precise, a slightly more general statement is proven in this section. Instead of merely deciding whether each p-value lies above or below a constant threshold, assume a modification of `MMCTest` is run which keeps on shrinking the confidence intervals of all p-values (placed vertically at a rank in $\{1, \ldots, m\}$ in a graphical representation) until they lie entirely within one of the *boxes*

$$C_j := \left[\frac{j\alpha}{m}, \frac{(j+1)\alpha}{m}\right)$$

for $j \in \{0, \ldots, m-1\}$ or within $C_m := [\alpha, 1]$, $\alpha < 1$. Depending on the p-values, none or more than one interval can be found to lie in one particular box.

A confidence interval lying entirely inside one of the boxes $C_j$ is stopped from being shrunk further. As all $C_j$ start and end at a point of the form $j\alpha/m$ for $j \in \{0, \ldots, m-1\}$ or at 1, knowing which box a confidence interval lies in implies knowing whether the corresponding p-value is above or below the threshold line (for the case of the Benjamini and Hochberg (1995) procedure). A similar construction with boxes placed at the discrete critical values of other testing procedures other than Benjamini and Hochberg (1995) is possible.

We assume that all intervals are two-sided Clopper and Pearson (1934) confidence intervals, the default choice for `MMCTest` given in Section A.1. While drawing samples, the length of a Clopper and Pearson (1934) confidence interval for each p-value goes down to zero (Section A.1). More precisely, its maximal length after having drawn $k$ samples is given in the next lemma.

**Lemma 2.15.** *The two-sided Clopper and Pearson (1934) confidence interval $I$ with coverage probability $1-\rho_k$ based on $k$ samples has maximal length $|I| \leq 2\sqrt{\frac{-1}{2k}\log(\rho_k)}$.*

The proof of Lemma 2.15 is identical to the first part of the proof of Lemma A.1.

Let $g(k) := 2\sqrt{\frac{-1}{2k}\log(\rho_k)}$. Define $D := \max_{j\in\{0,\dots,m\}} d(p,\overline{C}_j)$, where $d(p,\overline{C}_i)$ denotes the distance of a random $p$ to the complement of the interval $C_i$. Denote the sample mean of a p-value $p$ based on $k$ samples by $\hat{p}_k$. Suppose that $p \in C_j$ for a suitable $j \in \{0,\dots,m\}$.

If the distance of $p$ to $\hat{p}_k$ is less than $D/2$ and the confidence interval for $p$ (centered at $\hat{p}_k$) has a length less than $D/2$, the confidence interval for $p$ will fit into its box $C_j$. Therefore, sampling for $p$ will be stopped on reaching

$$\tau := \inf\{k : \ |\hat{p}_k - p| < D/2, g(k) < D/2\}.$$

Let $\tau_1,\dots,\tau_m$ be the stopping times of $m$ p-values as above and consider their order statistic $\tau_{(1)} \leq \dots \leq \tau_{(m)}$. The main statement of this section is the following theorem.

**Theorem 2.16.** *Consider $m > 2$ p-values independently distributed in $[0,1]$ according to a density which is bounded above by a finite constant. If $\log(\rho_k) = o(k^{2(\gamma+0.5)})$ for some $-\frac{1}{2} < \gamma < -\frac{1}{3}$, then $\mathbb{E}(\tau_{(m-s)}) < \infty$ for $s \geq 2$.*

*Proof of Theorem 2.16.* By assumption, the density of the p-values is bounded above by a constant $U$. Thus the density of $D$ is bounded above by $(m+1)U$.

In what follows, an upper bound on the survival function $\mathbb{P}(\tau > t)$ will be derived. The upper bound will then translate to an upper bound on the expectation of the order statistic of the $m$ random variables $\tau_1, \ldots, \tau_m$.

First, $\mathbb{P}(|\hat{p}_k - p| \geq D/2) \leq 2\exp(-\frac{1}{2}D^2 k)$ by Hoeffding's inequality (Hoeffding, 1963), where it was used that $\hat{p}_k = \frac{1}{k}\sum_{i=1}^{k} Y_i$ for $Y_i \sim \text{Bernoulli}(p)$ and $0 \leq Y_i \leq 1$.

Second, the event $\{g(k) < D/2\}$ is implied by $\{D > k^\gamma\}$ for sufficiently large $k$ for any $-\frac{1}{2} < \gamma < -\frac{1}{3}$. Indeed, $Dk^{-\gamma} > 1$ implies

$$g(k) = 2\sqrt{\frac{-1}{2k}\log(\rho_k)} < \frac{D}{2}k^{-\gamma-0.5}\frac{4}{\sqrt{2}}\sqrt{-\log(\rho_k)}.$$

Thus, using $\log(\rho_k) = o(k^{2(\gamma+0.5)})$, it follows that $g(k) < D/2$ for sufficiently large $k$.

The survival function $\mathbb{P}(\tau > t)$ can now be bounded above by conditioning on $D$. For large $t$,

$$\begin{aligned}
\mathbb{P}(\tau > t) &= \mathbb{P}(\tau > t | D \leq t^\gamma)\mathbb{P}(D \leq t^\gamma) + \mathbb{P}(\tau > t | D > t^\gamma)\mathbb{P}(D > t^\gamma) \\
&\leq (m+1)Ut^\gamma + \mathbb{P}(|\hat{p}_t - p| \geq D/2 \vee g(t) \geq D/2 | D > t^\gamma) \\
&\leq (m+1)Ut^\gamma + \mathbb{P}(|\hat{p}_t - p| \geq D/2 | D > t^\gamma) \\
&\leq (m+1)Ut^\gamma + 2\exp\left(-\frac{1}{2}t^{2\gamma+1}\right),
\end{aligned}$$

where $\mathbb{P}(\tau > t | D \leq t^\gamma) \leq 1$ was used and $\mathbb{P}(D \leq t^\gamma)$ was bounded using the bound on the density of $D$. Moreover, $\mathbb{P}(D > t^\gamma) \leq 1$ and $\tau > t$ if either $|\hat{p}_t - p| \geq D/2$ or $g(t) \geq D/2$. The latter vanishes in the limit as $\{D > t^\gamma\}$ implies $\{g(t) < D/2\}$. The upper bound $2\exp(-\frac{1}{2}D^2 t)$ on $\mathbb{P}(|\hat{p}_t - p| \geq D/2)$ has already been derived above, and $D > t^\gamma$ was then substituted. As $2\gamma + 1 > 0$, $2\exp\left(-\frac{1}{2}t^{2\gamma+1}\right)$ decays exponentially fast, hence $\mathbb{P}(\tau > t) \in O(t^\gamma)$.

Let $X_1, \dots, X_n$ be iid. random variables with cumulative distribution function (cdf) $F_X$ and let $r \in \{1, \dots, n\}$. Then the cdf of the $r$th order statistic can be expressed as $F_{X_{(r)}}(x) = \sum_{i=r}^n \binom{n}{i} F_X^i(x) \left(1 - F_X(x)\right)^{n-i}$ (David and Nagaraja, 2003).

Let $m \geq 3$ and let $r = m - s$, $s \geq 2$. Using the above expression for the $r$th order statistic, the expectation of $\tau_{(m-s)}$ can be bounded as

$$\begin{aligned}
\mathbb{E}(\tau_{(m-s)}) &= \int_0^\infty 1 - F_{\tau_{(m-s)}}(t)dt \\
&= \int_0^\infty \sum_{i=0}^{m-s-1} \binom{m}{i} F_\tau^i(t) \left(1 - F_\tau(t)\right)^{m-i} dt \\
&\leq \sum_{i=0}^{m-3} \binom{m}{i} \int_0^\infty \mathbb{P}(\tau > t)^{m-i} dt,
\end{aligned}$$

where $F_\tau(t) \leq 1$ was used and the fact that $s \geq 2$, hence $m - s - 1 \leq m - 3$. Using $\mathbb{P}(\tau > t) \in O(t^\gamma)$, the integrals $\int_0^\infty \mathbb{P}(\tau > t)^{m-i}dt$ behave like $\int_0^\infty t^{\gamma(m-i)}dt$ and converge as $\gamma < -1/3$ and $m \geq 3$ imply $\gamma(m-i) < -1$ for all $i \in \{0, \dots, (m-3)\}$. $\square$

As a consequence of the proof of Lemma A.1, the leading order term of the default spending sequence used for `MMCTest` (given at the beginning of Section A.1) is proportional to $k^{-2}$, hence Theorem 2.16 applies to it for any $-\frac{1}{2} < \gamma < -\frac{1}{3}$.

In particular, $\mathbb{E}(\tau_{(m-2)}) < \infty$ for $s = 2$. This means that `MMCTest`, in case hypotheses have not already been stopped earlier from receiving further samples, will determine in expected finite runtime for all but two confidence intervals which box $C_j$, $j \in \{1, \dots, m\}$, they lie in.

For testing using the Benjamini and Hochberg (1995) procedure, unfortunately, this result does not immediately allow to obtain a statement on the number of classified hypotheses (see Section 2.6.2). For the Bonferroni (1936) correction, however, determining which box a confidence interval lies in implies determining if the corresponding p-value lies above or below the threshold. As the threshold is constant, this immediately gives a decision on all but two hypotheses in finite expected runtime.

## 2.7 Discussion

We presented an open-ended sequential algorithm designed to implement multiplicity corrections for multiple Monte Carlo tests in the setting where the p-values are unknown and can only be approximated through simulation. In order to ensure repeatability and objectivity for Monte Carlo based multiple testing, we aim to compute the same classification as the one based on the p-values.

The main feature of `MMCTest` is that its output is guaranteed to be correct with a pre-specified probability, meaning that all its classifications are identical to the classifications based on the p-values.

Our simulation study shows that a complete classification can be computationally expensive, but that most hypotheses can be classified using a reasonable effort. For a realistic precision, `MMCTest` draws level with the performance of current methods which unlike `MMCTest` do not give a guarantee on their classifications being correct, such as the naive approach or the `MCFDR` algorithm. An ad-hoc variant of `MMCTest` outperforms the naive method and `MCFDR` both in terms of misclassifications and randomly classified hypotheses. Tuning the parameter $r$ of the spending sequence, spending all the remaining error probability or matching the effort exactly in the last iteration leaves scope for further research.

Conditions were identified which guarantee the bounded risk of classification errors and the convergence of the algorithm's output to the classification computed with the p-values. By verifying these conditions we showed that the `MMCTest` algorithm works for the Benjamini and Hochberg (1995) procedure as well as for the Bonferroni (1936) correction.

# 3 A Framework for Monte Carlo based Multiple Testing

## 3.1 Introduction

Using the same setting as the one considered in Chapter 2, this chapter aims to generalise the results of Chapter 2 into a framework for Monte Carlo based multiple testing. It thereby extends Chapter 2 in various ways: The framework allows to obtain theoretical guarantees for the test results of established algorithms and it gives theoretical bounds on the control of the false discovery rate. Moreover, it works with arbitrary step-up and step-down procedures and incorporates testing at a variable testing threshold.

Existing methods (Besag and Clifford, 1991; Lin, 2005; van Wieringen et al., 2008; Guo and Peddada, 2008; Sandve et al., 2011) return a set of rejected hypotheses which, at least to a certain extent, is random, where the randomness is coming from the Monte Carlo simulations and not from the underlying data. Consider the following example which will be revisited in Section 3.5. Sandve et al. (2011) use their method `MCFDR` to classify a genome dataset of Pekowska et al. (2010) with the aim to test if a gene modification appears more often in certain gene regions. Each gene region corresponds to one hypothesis. Sandve et al. (2011) report 2747 significant hypotheses out of 3466 hypotheses without providing guarantees on the stability of their finding. Recomputing the decisions on all hypotheses shows considerable variability: around 190 of the 3466 hypotheses are randomly classified in the sense that they switch from being rejected to non-rejected in more than 10% of all cases when repeatedly applying `MCFDR`. Conclusions based on the significances of these genes should therefore be questioned.

In the present chapter we will show how algorithms such as `MCFDR` can be modified to give a guarantee on the stability of their findings and thus how to reduce the Monte Carlo randomness in their results. Although in principle, increased stability can be achieved by augmenting the number of Monte Carlo samples, merely increasing the number of samples does not provide a guarantee on the decision of each hypothesis. The guarantees provided in the present chapter are effective for any number of samples.

The contribution of the chapter is threefold. First, Section 3.2 provides a framework for multiple hypothesis testing under the assumption that p-values are not available and thus have to be approximated using Monte Carlo methods. The framework is phrased as a generic algorithm which, under conditions, computes sub- and supersets of $h(p^*, \alpha^*)$ that converge to $h(p^*, \alpha^*)$ (Lemma 3.4 and Theorem 3.6 in Section 3.2.2). The framework also incorporates multiple testing at a (possibly unknown) corrected testing threshold, for instance using an estimate of the proportion of true null hypotheses.

Second, we show how to use the framework to modify established algorithms in such a way as to provide certain proven guarantees on their test results (Section 3.3).

Third, we simplify the condition on the multiple testing procedure in Section 3.4, yielding an easy-to-check criterion for an arbitrary step-up or step-down procedure (Section 3.4.1). We then use the simplified criterion to show that many widely used procedures can be employed in our framework (Section 3.4.2).

One specific implementation of our generic algorithm is the `MMCTest` algorithm. In contrast to the present chapter which presents results for a generic algorithm and a generic multiple testing procedure, `MMCTest` focuses on one specific implementation only as well as on the two specific multiple testing procedures of Bonferroni (1936) and Benjamini and Hochberg (1995). The present chapter therefore extends the previous results by showing that a correct test result can also be obtained through appropiate modifications of existing methods. Moreover, hypothesis testing at a variable testing threshold is not possible with `MMCTest`, and the previous chapter did not provide a simple criterion to prove whether an arbitrary step-up or step-down procedure allows one to classify hypotheses without knowledge of the p-values.

In Section 3.5 we pick up our discussion of the biological dataset of Pekowska et al. (2010). We show that our proposed modifications can easily be implemented in practice, come at virtually no additional computational cost and lead to a certain way of reporting multiple testing results as three sets together with an error bound on their correctness.

The chapter concludes with a discussion in Section 3.6. Selected insightful proofs are given in the chapter, lengthy and technical proofs can be found in Section B.1.

Throughout the chapter, let $|\cdot|$ denote the length of an interval or the size of a set. Let $\|\cdot\|$ denote the Euclidean norm. For an interval $I \subset \mathbb{R}$, let $\min I$ and $\max I$

denote its lower and upper limit, respectively. For any set $S \subseteq \{1, \ldots, m\}$, where $m \in \mathbb{N}$, let $S^c$ denote the complement of $S$ with respect to $\{1, \ldots, m\}$. We abbreviate $(x_1, \ldots, x_n)$ by $x_{1:n}$, where $x_{1:0} = \emptyset$.

## 3.2 The framework

Our framework includes two components, the multiple testing procedure $h$ and a generic algorithm presented in Section 3.2.1. Combining both the testing procedure and the algorithm yields a framework which, under conditions, guarantees the correctness of its test result (Section 3.2.2).

### 3.2.1 The generic algorithm

We propose to use the following generic sequential algorithm to draw samples for each hypothesis. As the p-values are unknown, in each iteration $n$, the generic algorithm computes intervals $I_n^i$ for each $p_i^*$, $i \in \{1, \ldots, m\}$, as well as an interval $I_n^{m+1}$ for $\alpha^*$. Usually, these intervals will be confidence intervals, in which case our algorithm will compute sub- and supersets of $h(p^*, \alpha^*)$ (Section 3.2.2).

Although the threshold will be, in most cases, a function $\alpha^* = g(p^*)$ of the p-values (see Section 3.3), it is sensible to not restrict the multiple testing procedure to $h(p^*) = h(p^*, g(p^*))$ and to keep a separate interval $I_n^{m+1}$ for $\alpha^*$ instead: Naturally, one could use confidence bounds on $p^*$ to obtain a plug-in interval for $\alpha^*$ (provided that $g$ is monotonic). However, Example 3.10 demonstrates that for the testing threshold of Pounds and Cheng (2006), Hoeffding's inequality (Hoeffding, 1963) allows one to construct a tighter confidence interval for $\alpha^*$ than the plug-in interval, thus yielding a faster convergence to $h(p^*, \alpha^*)$ as well as considerably more decisions on individual hypotheses in a real-data study (Section 3.5).

The generic algorithm draws Monte Carlo samples in each iteration $n$, denoted by the observations $O_n$. These are typically sets of samples drawn for all hypotheses or for a subset of the hypotheses. The decision which observations to sample may depend on the history of observations drawn up to iteration $n-1$.

**Algorithm 3.1:** Generic algorithm

```
1 $\underline{A}_0 \leftarrow \emptyset$; $\overline{A}_0 \leftarrow \{1,\dots,m\}$; $I_0^i \leftarrow [0,1]$, $i \in \{1,\dots,m\}$, $I_0^{m+1} = \mathbb{R}$;
2 for $n = 1, 2, \dots$ do
3     Choose which $O_n$ to sample based on $O_{1:n-1}$;
4     Sample $O_n$;
5     $I_n^i \leftarrow F_i(O_{1:n}) \cap I_{n-1}^i$, $i \in \{1,\dots,m+1\}$;
6     $\overline{A}_n \leftarrow h((\min I_n^i)_{i\in\{1,\dots,m\}}, \max I_n^{m+1})$;
7     $\underline{A}_n \leftarrow h((\max I_n^i)_{i\in\{1,\dots,m\}}, \min I_n^{m+1})$;
```

In each iteration $n$, Algorithm 3.1 uses the history of samples observed up to iteration $n-1$ to determine a new set of observations $O_n$ to be sampled. The key idea of Algorithm 3.1 is to apply the multiple testing procedure $h$ to lower ($\min I_n^i$) and upper ($\max I_n^i$) confidence limits of the $(I_n^i)_{n\in\mathbb{N}}$, $i \in \{1,\dots,m\}$. This yields two sets $\overline{A}_n$ and $\underline{A}_n$. Each of these confidence intervals $I_n^i$ is computed by a function $F_i$, $i \in \{1,\dots,m\}$, using the current history of observations $O_{1:n}$, $n \in \mathbb{N}$. For generality, we do not impose that $F_i$ computes any specific type of confidence interval. Intersecting the intervals produces a nested sequence of $(I_n^i)_{n\in\mathbb{N}}$ for each $i \in \{1,\dots,m\}$.

**Example 3.2** (Improved naive method)**.** *A widely used method in practice to estimate $h(p^*, \alpha^*)$ is to draw a constant number of samples $s$ for each hypothesis $H_{0i}$, where $i \in \{1,\dots,m\}$, then compute a point estimate of each p-value and classify all hypotheses at a constant threshold $\alpha^*$ based on these point estimates (Nusinow et al., 2012; Gusenleitner et al., 2012; Rahmatallah et al., 2012; Zhou et al., 2013; Li et al.,* *2012; Cohen et al., 2012). We will call this the naive method. The naive method can be applied to any multiple testing procedure $h$.*

*In the following we present an improvement of the naive method by stating a concrete implementation of Algorithm 3.1. As shown in Section 3.3, under conditions on $h$, the sets $\underline{A}_n$ ($\overline{A}_n$) defined in Algorithm 3.1 will be subsets (supersets) of $h(p^*, \alpha^*)$ in each iteration $n$ of our improved naive method up to a pre-specified error probability.*

*One key ingredient of the improved naive method are the confidence sequences given in Lai (1976): for independent $Y_1, Y_2, \ldots \sim$ Bernoulli($p$),*

$$\mathbb{P}(g_n^\beta(S_n) < p < f_n^\beta(S_n)\ \forall n \geq 1) \geq 1 - \beta,$$

*where $S_n = \sum_{i=1}^n Y_i$ and $g_n^\beta(x) < f_n^\beta(x)$ are the two distinct (Lai, 1976) roots of $(n+1)\binom{n}{x}p^x(1-p)^{n-x} = \beta$ for a given $\beta \in (0,1)$.*

*In many applications of the naive method, multiple tests are based on a test statistic and it is possible to sample under the null hypothesis. Let $X_n^i = 1$ if the test statistic evaluated on the $n^{th}$ sample drawn for hypothesis $H_{0i}$ exceeds the observed test statistic, otherwise $X_n^i = 0$. For our improved naive method, we draw one new sample per hypothesis in each iteration $n$.*

*The improved naive method is obtained by defining*

$$\begin{aligned} O_n &= (X_n^1, \ldots, X_n^m), \\ F_i(O) &= \left[ g_{|O|}^\beta \left( \sum_{j=1}^{|O|} O_j^i \right), f_{|O|}^\beta \left( \sum_{j=1}^{|O|} O_j^i \right) \right],\ i = 1, \ldots, m, \\ F_{m+1}(O) &= \{\alpha^*\}, \end{aligned}$$

*where $O^i_j = X^i_j$ and $|O_{1:n}| = n$.*

*Although the above method is open-ended, we usually stop the improved naive method after a pre-specified total number of iterations $s$. In this case, solely the two test results in $\underline{A}_s$ and $\overline{A}_s$ based on the intervals of the last iteration will be returned as result of the algorithm.*

In Example 3.2, the testing threshold $\alpha^*$ is assumed to be constant. However, the interval $F_{m+1}(O_{1:n})$ for $\alpha^*$ is needed if $\alpha^*$ depends on $p^*$. For instance, this is the case for thresholds depending on an estimate of the proportion of true null hypotheses which is usually a functional of $p^*$. Using such an estimated threshold potentially results in more significant hypotheses which is desired in practice.

Starting with the work of Schweder and Spjøtvoll (1982), many authors have investigated estimators of the proportion of true null hypotheses, such as Benjamini and Hochberg (2000), Storey (2002), Langaas et al. (2005), Cheng (2006), Pounds and Cheng (2006), Jiang and Doerge (2008), Finner and Gontscharuk (2009), Hwang (2011).

### 3.2.2 Convergence results

This section states our main results. For this we need the following extension of a monotonic multiple testing procedure considered first in Section 2.2:

**Definition 3.3.** *$h$ is monotonic if $h(p, \alpha) \subseteq h(q, \alpha')$ for $p \geq q$ and $\alpha \leq \alpha'$.*

A multiple testing procedure is thus monotonic if smaller p-values (as introduced in Tamhane and Liu, 2008) or a higher testing threshold (see Roth, 1999) lead to more rejections.

Suppose in each iteration $n \in \mathbb{N}$, each p-value $p_i^*$ is contained in its interval $F_i(O_{1:n})$, $i \in \{1, \dots, m\}$, and the testing threshold $\alpha^*$ is contained in the interval $F_{m+1}(O_{1:n})$, expressed as the event

$$R_1 = \{\alpha^* \in F_{m+1}(O_{1:n}), p_i^* \in F_i(O_{1:n}) \; \forall i \in \{1, \dots, m\}, n \in \mathbb{N}\}.$$

The following lemma shows that on the event $R_1$, classifying hypotheses based on upper and lower interval bounds allows Algorithm 3.1 to compute sub- and supersets of $h(p^*, \alpha^*)$ for monotonic multiple testing procedures $h$. The lemma extends a similar result for `MMCTest` (Lemma 2.3) to a variable testing threshold.

**Lemma 3.4.** *Let $h$ be a monotonic multiple testing procedure. Then,*

1. $\underline{A}_n \nearrow$ *and* $\overline{A}_n \searrow$ *as* $n \to \infty$,
2. $\underline{A}_n \subseteq h(p^*, \alpha^*) \subseteq \overline{A}_n \; \forall n \in \mathbb{N}$ *on the event* $R_1$.

*Proof.* 1. By construction, Algorithm 3.1 computes nested intervals, thus $\overline{p}_n = (\max I_n^i)_{i \in \{1,\dots,m\}} \searrow$ and $\overline{\alpha}_n = \max I_n^{m+1} \searrow$ as well as $\underline{p}_n = (\min I_n^i)_{i \in \{1,\dots,m\}} \nearrow$ and $\underline{\alpha}_n = \min I_n^{m+1} \nearrow$. Hence,

$$\underline{A}_n = h(\overline{p}_n, \underline{\alpha}_n) \subseteq h(\overline{p}_{n+1}, \underline{\alpha}_n) \subseteq h(\overline{p}_{n+1}, \underline{\alpha}_{n+1}) = \underline{A}_{n+1},$$
$$\overline{A}_n = h(\underline{p}_n, \overline{\alpha}_n) \supseteq h(\underline{p}_{n+1}, \overline{\alpha}_n) \supseteq h(\underline{p}_{n+1}, \overline{\alpha}_{n+1}) = \overline{A}_{n+1},$$

where the first (second) subset relation follows from the monotonicity of $h$ in the first (second) argument.

2. On the event $R_1$, $p_i^* \in I_n^i$ and $\alpha^* \in I_n^{m+1}$ for all $i$ and $n$, thus $\overline{p}_n \geq p_n^* \geq \underline{p}_n$ and $\underline{\alpha}_n \leq \alpha^* \leq \overline{\alpha}_n$. By monotonicity of $h$, $\underline{A}_n = h(\overline{p}_n, \underline{\alpha}_n) \subseteq h(p^*, \alpha^*) \subseteq h(\underline{p}_n, \overline{\alpha}_n) = \overline{A}_n$

$\forall n \in \mathbb{N}$. □

The first part of Lemma 3.4 is not dependent on the event $R_1$. It follows purely from the construction of Algorithm 3.1 which computes nested intervals for each $p_i^*$, $i \in \{1, \ldots, m\}$. The second part of Lemma 3.4 shows that on $R_1$, in any iteration $n$, all the hypotheses in the set $\underline{A}_n$ ($\overline{A}_n^c$) can already be classified as being rejected (non-rejected).

Additional properties of Algorithm 3.1 can be derived for any monotonic multiple testing procedure $h$ and choice of $p^*$, $\alpha^*$ which satisfy the following condition.

**Condition 3.5.** *1. Let $p, q \in [0,1]^m$ and $\alpha \in \mathbb{R}$. If $q_i \leq p_i$ $\forall i \in h(p, \alpha)$ and $q_i \geq p_i$ $\forall i \notin h(p, \alpha)$, then $h(p, \alpha) = h(q, \alpha)$.*

*2. There exists $\delta > 0$ such that $p \in [0,1]^m$, $\alpha \in [0,1]$ and $\|p - p^*\| \vee |\alpha - \alpha^*| < \delta$ imply $h(p, \alpha) = h(p^*, \alpha^*)$.*

Condition 3.5 ensures that lowering (increasing) the p-value of any rejected (non-rejected) hypothesis does not affect the result of $h$. Moreover, we require that there exists a neighbourhood of $p^*$ and $\alpha^*$ on which $h$ is constant. In Section 3.4 we will simplify Condition 3.5 for so-called step-up and step-down procedures.

Condition 3.5 again is a generalisation of a similar condition seen in Chapter 2 for `MMCTest`: the first part is identical to Condition 2.4 whereas the second part now requires the difference of the p-values $p$ and $p^*$ as well as of $\alpha$ and $\alpha^*$ to be less than the constant $\delta$.

We will call a monotonic multiple testing procedure $h$ *admissible* for $p^*$ and $\alpha^*$ if it satisfies Condition 3.5. The multiple testing procedures we consider in this chapter (see Section 3.4) are admissible for all but a null set of $p^*$ and $\alpha^*$ (with respect to the Lebesgue measure).

A second condition is necessary to obtain convergence of the two bounds $\underline{A}_n$ and $\overline{A}_n$ established in Lemma 3.4 to $h(p^*, \alpha^*)$ as $n \to \infty$. Whereas on the event $R_1$, all hypotheses in $\underline{A}_n$ ($\overline{A}_n^c$) can already be rejected (non-rejected), we additionally require that the length of each interval belonging to a yet unclassified hypothesis in the set $\overline{A}_n \setminus \underline{A}_n$ or to the threshold goes to zero:

$$R_2 = \left\{ \max\{|F_i(O_{1:n})| : i \in \overline{A}_n \setminus \underline{A}_n \cup \{m+1\}\} \to 0 \text{ as } n \to \infty \right\}.$$

The following theorem improves upon Lemma 3.4 on the more restrictive event $R = R_1 \cap R_2$ and thus extends Theorem 2.6 proven for the `MMCTest` algorithm to the situation of variable testing thresholds:

**Theorem 3.6.** *Let $h$ be an admissible multiple testing procedure for $p^*$ and $\alpha^*$. On the event $R$, both sequences $(\underline{A}_n)_{n\in\mathbb{N}}$ and $(\overline{A}_n)_{n\in\mathbb{N}}$ converge to $h(p^*, \alpha^*)$, i.e. there exists $n_0 \in \mathbb{N}$ such that $\underline{A}_n = h(p^*, \alpha^*) = \overline{A}_n$ $\forall n \geq n_0$.*

*Proof.* Let $\overline{\alpha}_n = \max I_n^{m+1}$, $\underline{\alpha}_n = \min I_n^{m+1}$ as well as $B_n = \overline{A}_n \setminus \underline{A}_n$. Suppose $\exists i \in \limsup_{n\to\infty} B_n$. On the event $R_2$, $|I_n^i| \to 0$ as $n \to \infty$ for $i \in \limsup_{n\in\mathbb{N}} B_n$ as well as $|I_n^{m+1}| \to 0$ as $n \to \infty$. Let $\delta$ be as given in Condition 3.5. As $B_n \subseteq \{1, \ldots, m\}$ is finite $\forall n \in \mathbb{N}$, there exists $n_0 \in \mathbb{N}$ such that $|I_n^i|^2 < \delta^2/m$ and $|\underline{\alpha}_n - \overline{\alpha}_n| < \delta$ for $n \geq n_0$ and all $i \in \limsup_{n\to\infty} B_n$.

We show that for all $n \geq n_0$,

$$\overline{A}_n = h((\min I_n^i)_{i\in\{1,\ldots,m\}}, \overline{\alpha}_n) = h(p^*, \alpha^*) = h((\max I_n^i)_{i\in\{1,\ldots,m\}}, \underline{\alpha}_n) = \underline{A}_n.$$

To do this, we show

$$h(p^{(1)}, \overline{\alpha}_n) = h(p^{(2)}, \overline{\alpha}_n) = h(p^{(3)}, \overline{\alpha}_n),$$
$$h(p^{(3)}, \overline{\alpha}_n) = h(p^{(4)}, \alpha^*) = h(p^{(5)}, \underline{\alpha}_n),$$
$$h(p^{(5)}, \underline{\alpha}_n) = h(p^{(6)}, \underline{\alpha}_n) = h(p^{(7)}, \underline{\alpha}_n),$$

where

$$p^{(1)} := (\min I_n^i)_{i\in\{1,\dots,m\}}, \qquad p^{(4)} := p^*,$$
$$p^{(2)} := \begin{cases} \min I_n^i & i \in \overline{A}_n, \\ p_i^* & i \notin \overline{A}_n, \end{cases} \qquad p^{(5)} := \begin{cases} \max I_n^i & i \in B_n, \\ p_i^* & i \notin B_n, \end{cases}$$
$$p^{(3)} := \begin{cases} \min I_n^i & i \in B_n, \\ p_i^* & i \notin B_n, \end{cases} \qquad p^{(6)} := \begin{cases} \max I_n^i & i \in \overline{A}_n, \\ p_i^* & i \notin \overline{A}_n, \end{cases}$$

and $p^{(7)} := (\max I_n^i)_{i\in\{1,\dots,m\}}$. The following holds true on the event $R_1$.

(1) By definition, $\overline{A}_n = h(p^{(1)}, \overline{\alpha}_n)$. As $p_j^{(2)} = p_j^* \geq \min I_n^j = p_j^{(1)}$ $\forall j \notin \overline{A}_n$ and $p_j^{(2)} = p_j^{(1)}$ $\forall j \in \overline{A}_n$, the first part of Condition 3.5 yields $\overline{A}_n = h(p^{(1)}, \overline{\alpha}_n) = h(p^{(2)}, \overline{\alpha}_n)$ for a fixed $\overline{\alpha}_n$.

(2) As $(\max I_n^i)_{i\in\{1,\dots,m\}} \geq p^{(3)}$ and because $h$ is monotonic, $\underline{A}_n \subseteq h(p^{(3)}, \underline{\alpha}_n) \subseteq h(p^{(3)}, \overline{\alpha}_n)$. As $p_j^{(2)} = \min I_n^j \leq p_j^* = p_j^{(3)}$ $\forall j \in \underline{A}_n$ and $p_j^{(2)} = p_j^{(3)}$ $\forall j \notin \underline{A}_n$, the first part of Condition 3.5 yields $h(p^{(2)}, \overline{\alpha}_n) = h(p^{(3)}, \overline{\alpha}_n)$.

(3) On the event $R_1$, $|\underline{\alpha}_n - \overline{\alpha}_n| < \delta$ implies $|\alpha^* - \overline{\alpha}_n| < \delta$ and $|I_n^i|^2 < \delta^2/m$ implies $\|p^{(3)} - p^*\| < \delta$. The second part of Condition 3.5 thus yields $h(p^{(3)}, \overline{\alpha}_n) = h(p^{(4)}, \alpha^*) = h(p^*, \alpha^*)$ $\forall n \geq n_0$.

Arguing similarly to (1), (2), (3) we can show $h(p^{(4)}, \alpha^*) = h(p^{(5)}, \underline{\alpha}_n)$ as well as $h(p^{(5)}, \underline{\alpha}_n) = h(p^{(6)}, \underline{\alpha}_n)$ and $h(p^{(6)}, \underline{\alpha}_n) = h(p^{(7)}, \underline{\alpha}_n) = \underline{A}_n$. □

In the next section, we will use Lemma 3.4 and Theorem 3.6 to establish guarantees on the test result of existing algorithms.

Suppose Algorithm 3.1 is used in connection with an admissible multiple testing procedure controlling the familywise error rate (fwer). Then at any stage, the fwer is also controlled for all the rejections in $\underline{A}_n \subseteq h(p^*, \alpha^*)$. This is easily proven using Boole's inequality.

A similar statement, however, is not true for admissible multiple testing procedures controlling the false discovery rate (fdr). Although the fdr is not generally controlled for subsets $\underline{A}_n \subseteq h(p^*, \alpha^*)$ or supersets $\overline{A}_n \supseteq h(p^*, \alpha^*)$, the following guarantees hold if Algorithm 3.1 is run with suitable stopping times.

**Lemma 3.7.** *Let $h$ control the fdr at level $\alpha$, let $\underline{V}_n$ ($\overline{V}_n$) be the set of rejected true null hypotheses in $\underline{A}_n$ ($\overline{A}_n$) for $n \in \mathbb{N}$ and let $\eta \geq 1$, $\xi \geq 0$.*

1. *$\mathbb{E}\left(|\underline{V}_s|/|\underline{A}_s|\right) \leq \eta\alpha$ for the stopping time $s = \min\{n \in \mathbb{N} : |\overline{A}_n|/|\underline{A}_n| \leq \eta\}$.*

2. *$\mathbb{E}\left(|\overline{V}_t|/|\overline{A}_t|\right) \leq \alpha + \xi$ for $t = \min\{n \in \mathbb{N} : (|\overline{A}_n| - |\underline{A}_n|)/|\overline{A}_n| \leq \xi\}$.*

*Proof.* Let $R$ be the set of rejected hypotheses and $V$ be the set of rejected true null hypotheses. As $\underline{A}_n \subseteq R \subseteq \overline{A}_n$ for all $n \in \mathbb{N}$, $|\underline{A}_n| \leq |R| \leq |\overline{A}_n|$. Moreover, as $\underline{A}_n \subseteq R$, all rejected true null hypotheses in $\underline{A}_n$ are also in $R$ and thus by definition also in $V$, hence $\underline{V}_n \subseteq \underline{A}_n \subseteq R$ implies $|\underline{V}_n| \leq |V|$. As the difference in numbers of rejected true null hypotheses in $V$ and $\overline{V}_n$ cannot differ by more than the number of undecided hypotheses $|\overline{A}_n| - |\underline{A}_n|$ for any $n \in \mathbb{N}$ it follows that $|V| \leq |\overline{V}_n| \leq |V| + (|\overline{A}_n| - |\underline{A}_n|)$.

1. Using the above,

$$\frac{|\underline{V}_n|}{|\underline{A}_n|} = \frac{|V|}{|R|} + \frac{|\underline{V}_n||R| - |\underline{A}_n||V|}{|\underline{A}_n||R|} \le \frac{|V|}{|R|} + \frac{|V|}{|R|}\frac{|R| - |\underline{A}_n|}{|\underline{A}_n|} \le \frac{|V|}{|R|}\frac{|R|}{|\underline{A}_n|} \le \frac{|V|}{|R|}\frac{|\overline{A}_n|}{|\underline{A}_n|}$$

for all $n \in \mathbb{N}$, thus $\mathbb{E}(|\underline{V}_s|/|\underline{A}_s|) \le \eta\mathbb{E}(|V|/|R|) = \eta\alpha$.

2. Similarly,

$$\frac{|\overline{V}_n|}{|\overline{A}_n|} \le \frac{|V| + (|\overline{A}_n| - |\underline{A}_n|)}{|\overline{A}_n|} \le \frac{|V|}{|R|} + \frac{|\overline{A}_n| - |\underline{A}_n|}{|\overline{A}_n|}$$

for all $n \in \mathbb{N}$, thus $\mathbb{E}(|\overline{V}_t|/|\overline{A}_t|) \le \mathbb{E}(|V|/|R|) + \xi = \alpha + \xi$.

□

In Lemma 3.7, we define the fraction in the definition of the stopping time $s$ (time $t$) to be zero if $|\underline{A}_n|$ ($|\overline{A}_n|$) is zero as in this case, false rejection errors are impossible. Lemma 3.7 thus provides two different guarantees on the fdr, a multiplicative one on the set of rejected hypotheses $\underline{A}_s$ and an additive guarantee on the rejections in $\overline{A}_t$ with respect to the two stopping times $s$ and $t$.

## 3.3 Improving existing algorithms

In this section we introduce a class of established methods which estimate $h(p^*, \alpha^*)$ and show how the framework given by Algorithm 3.1 can be used to modify these methods in such a way as to provide a guarantee on the correctness of their test results. We will demonstrate our proposed modifications by extending the improved naive method presented in Example 3.2 to the situation of an estimated testing threshold.

Consider an existing method to compute $h(p^*, \alpha^*)$. The threshold $\alpha^*$ can either be constant or given by a monotonic (increasing or decreasing) function $g : [0, 1]^m \to \mathbb{R}$, thus $\alpha^* = g(p^*)$. In the latter case, $\alpha^*$ is a function of $p^*$ and thus unknown itself.

Methods working with bootstrap point estimates of $p^*$ (Besag and Clifford, 1991; van Wieringen et al., 2008; Sandve et al., 2011; Jiang and Salzman, 2012), fitted distributions (Knijnenburg et al., 2009) or permutation based methods (Westfall and Young, 1993; Westfall and Troendle, 2008; Meinshausen, 2006) can be phrased in the following way: Draw independent samples $X_{ij} \sim \text{Bernoulli}(p_i^*)$, $j \in \mathbb{N}$, for each $i \in \{1, \dots, m\}$. Use a finite number $S_i$ of these samples $X_{i1}, \dots, X_{i,S_i}$ to compute a p-value estimate $\hat{p}_i$ of $p_i^*$, where $S_i$ is a (random) index and $i \in \{1, \dots, m\}$. Estimate the testing threshold $\alpha^*$ using the plug-in estimate $\hat{\alpha} = g(\hat{p})$, where $\hat{p} = (\hat{p}_1, \dots, \hat{p}_m)$. Return $h(\hat{p}, \hat{\alpha})$ as the test result.

Based on Algorithm 3.1 we propose to modify any method of the above type by

1. Maintaining a confidence sequence (Lai, 1976) with a coverage probability of $1 - \epsilon/m$ for each p-value $p_i^*$, $i \in \{1, \dots, m\}$, and by using each sequence as $F_i(O_{1:n})$ in Algorithm 3.1. The overall error probability $\epsilon$ is chosen by the user.

2. Computing plug-in bounds $F_{m+1}(O_{1:n})$ for $\alpha^*$ using the monotonicity of $g$ and the above confidence sequences.

3. Reporting hypotheses in $\underline{A}_n$ as rejected and in $\overline{A}_n^c$ as non-rejected. The remaining hypotheses are still undecided.

As the confidence sequence of Lai (1976) satisfies $\mathbb{P}(\exists n : p_i^* \notin F_i(O_{1:n})) < \beta$ for each $p_i^*$ (see Example 3.2), the choice $\beta = \epsilon/m$ yields

$$\mathbb{P}(\exists i, n : p_i^* \notin F_i(O_{1:n})) \leq \sum_{i=1}^{m} \mathbb{P}(\exists n : p_i^* \notin F_i(O_{1:n})) \leq \sum_{i=1}^{m} \epsilon/m = \epsilon,$$

and hence $\mathbb{P}(p_i^* \in F_i(O_{1:n})\ \forall i \in \{1, \dots, m\}, n \in \mathbb{N}) \geq 1 - \epsilon$. The event $R_1$ thus occurs with probability at least $1 - \epsilon$.

Consequently, any modified method of the above type has the following advantage over its unimproved counterpart:

**Remark 3.8.** *By Lemma 3.4, a modified method of the above type has the property that all the hypotheses in the set $\underline{A}_n$ ($\overline{A}_n^c$) which are rejected (non-rejected) in any iteration $n$ are indeed correctly rejected (non-rejected) with probability at least $1 - \epsilon$.*

Remark 3.8 applies to the improved naive method (Example 3.2) upon stopping in iteration $s$ as well as to the methods presented in the following two examples. First, we generalise Example 3.2 to the situation where the testing threshold is unknown.

**Example 3.9.** *Additionally to the setting of Example 3.2, we assume that multiple testing is carried out at the corrected testing threshold $\alpha^* = t^*/\hat{\pi}_0(p^*)$, where $t^*$ is an uncorrected threshold (typically $t^* = 0.05$ or $t^* = 0.1$) and $\hat{\pi}_0(p) = \min\left(1, \frac{2}{m}\sum_{i=1}^m p_i\right)$ is an estimator of the proportion of true null hypotheses (Pounds and Cheng, 2006). Recent applications of this threshold include Han and Dalal (2012), Lu et al. (2011), Jupiter et al. (2010), Cheng (2009).*

*As $\hat{\pi}_0(p^*)$ depends on the p-values, the corrected threshold $\alpha^*$ is unknown in practice. We thus need to compute a confidence interval for it. The interval can be constructed using the monotonicity of $\hat{\pi}_0(p)$: in iteration $n$, $\underline{\pi}_n = \hat{\pi}_0(\min I_n^1, \dots, \min I_n^m)$ is a lower bound on $\hat{\pi}_0(p^*)$, likewise $\overline{\pi}_n = \hat{\pi}_0(\max I_n^1, \dots, \max I_n^m)$ is an upper bound. This immediately translates to the interval $F_{m+1}(O_{1:n}) = [t^*/\overline{\pi}_n, t^*/\underline{\pi}_n]$ for $\alpha^*$.*

We try to improve Example 3.9 by using a (hopefully) tighter confidence interval $F_{m+1}(O_{1:n})$ for $\alpha^*$ based on Hoeffding's inequality (Hoeffding, 1963).

**Example 3.10.** *Suppose we have observed $s$ samples $X_1^i, \ldots, X_s^i$ per hypothesis $H_{0i}$, where $X_j^i$ is the indicator of an exceedance for the $j$th sample drawn for $H_{0i}$ as in Example 3.2. Then for all $u > 0$, by Hoeffding's inequality:*

$$\mathbb{P}\left(\left|\frac{1}{ms}\sum_{i=1}^{m}\sum_{j=1}^{s} X_j^i - \frac{1}{m}\sum_{i=1}^{m} p_i^*\right| \geq u\right) \leq 2\exp\left(-2msu^2\right).$$

*Thus for a given $\eta \in [0,1]$,*

$$\frac{1}{ms}\sum_{i=1}^{m}\sum_{j=1}^{s} X_j^i \pm \sqrt{-\frac{1}{2ms}\log\left(\frac{\eta}{2}\right)}$$

*are boundaries of a $1-\eta$ confidence interval for $\frac{1}{m}\sum_{i=1}^{m} p_i^*$. Using the monotonicity of the mapping $x \mapsto t^*/\min(1,2x)$, this immediately translates to a $1-\eta$ confidence interval for $\alpha^*$.*

*When using Hoeffding's interval in the improved naive method, we allocate an error of $\eta = \epsilon/(m+1)$ to the computation of Hoeffding's interval as well as to the computation of each of the $m$ confidence sequences for the p-values. As the improved naive method is open-ended, we use a non-negative real sequence $(\eta_n)_{n\in\mathbb{N}}$ satisfying $\sum_{n=1}^{\infty}\eta_n = \eta$ to distribute $\eta$ for Hoeffding's interval over all iterations of the algorithm, thus computing it at level $\eta_n$ in each iteration $n$.*

Both the plug-in interval (Example 3.9) and Hoeffding's confidence interval (Example 3.10) will be evaluated in Section 3.5.

## 3.4 Admissibility of step-up and step-down procedures

Although the multiple testing procedure $h$ does not have to be of a special form, many procedures used in practice such as the ones of Bonferroni (1936), Sidak (1967), Holm (1979), Hochberg (1988) or the one of Benjamini and Hochberg (1995) belong to a certain class of procedures, called step-up and step-down procedures. We will simplify Condition 3.5 for step-up and step-down procedures in Section 3.4.1 and use the simplified condition in Section 3.4.2 to verify that many widely used procedures are admissible. As shown in Section B.2, the Hommel (1988) procedure is an example of a procedure which is not admissible.

### 3.4.1 A simplified admissibility condition

Suppose we are given an arbitrary step-up procedure $h_u$ or step-down procedure $h_d$ (Romano and Shaikh, 2006) returning the set of rejected indices. For our purposes, we phrase these two procedures in terms of a threshold function $\tau_\alpha : \{1, \dots, m\} \to [0, 1]$ which depends on a threshold $\alpha \in [0, 1]$ and returns the critical value $\tau_\alpha(i)$ each $p_{(i)}$ is compared to:

$$h_u(p, \alpha) = \left\{ i \in \{1, \dots, m\} : p_i \leq \max_{j \in \{1,\dots,m\}} \{p_{(j)} : p_{(j)} \leq \tau_\alpha(j)\} \right\}, \tag{3.1}$$

$$h_d(p, \alpha) = \left\{ i \in \{1, \dots, m\} : p_i < \min_{j \in \{1,\dots,m\}} \{p_{(j)} : p_{(j)} > \tau_\alpha(j)\} \right\}, \tag{3.2}$$

where $\max \emptyset := 0$, $\min \emptyset := 1$, and where the order statistic of $p_1, \dots, p_m$ is denoted by $p_{(1)} \leq \dots \leq p_{(m)}$.

We assume that the threshold function $\tau_\alpha$ satisfies the following condition.

**Condition 3.11.** *1. $\tau_\alpha(i)$ is non-decreasing in $i$ for each fixed $\alpha$.*

*2. $\tau_\alpha(i)$ is continuous in $\alpha$ and non-decreasing in $\alpha$ for each fixed $i$.*

By the following lemma, a step-up or step-down procedure is admissible if the threshold function $\tau_\alpha$ defining it satisfies Condition 3.11.

**Lemma 3.12.** *If $\tau_\alpha$ satisfies Condition 3.11 then the corresponding $h_u$ and $h_d$ are monotonic and satisfy the first part of Condition 3.5. If moreover $\tau_{\alpha^*}(i) \neq p^*_{(i)}$ for all $i \in \{1, \dots, m\}$, $h_u$ and $h_d$ also satisfy the second part of Condition 3.5 for $p^*$ and $\alpha^*$.*

**Remark 3.13.** *Suppose the p-values $p^*$ are random with a distribution that is absolutely continuous with respect to the Lebesgue measure. Then, for a fixed $\alpha^*$, the p-values not satisfying the conditions of Lemma 3.12 form a null set.*

### 3.4.2 Examples of admissible step-up and step-down procedures

This section shows that a variety of commonly used step-up and step-down procedures are monotonic and satisfy Condition 3.11.

The following multiple testing procedures are determined by $\tau_\alpha(i)$, where $i \in \{1, \dots, m\}$, and control the fwer or the fdr at a threshold $\alpha$. We denote the hypothesis corresponding to the ordered p-value $p_{(i)}$ by $H_{0(i)}$, $i \in \{1, \dots, m\}$.

In most cases, Condition 3.11 can be checked by considering the derivatives of $\tau_\alpha(i)$ with respect to $\alpha$ and $i$, thus regarding $i$ as a continuous parameter. Unless stated otherwise, all the threshold functions listed below are clearly non-decreasing in both $i$ and $\alpha$ as well as continuous in $\alpha$ and thus satisfy Condition 3.11.

The Bonferroni (1936) correction can be derived from either a step-up or a step-down procedure using the constant threshold function $\tau_\alpha(i) = \alpha/m$.

The following step-up procedures are admissible:

1. The Simes (1986) procedure rejects $\cap_{i\in\{1,\dots,m\}}H_{0i}$ if there exists $k \in \{1,\dots,m\}$ such that $p_{(k)} \leq k\alpha/m$. It can be used in our framework with the help of the following modification: Once $h_u(p,\alpha)$ for a step-up procedure with threshold function $\tau_\alpha(i) = i\alpha/m$ is correctly determined, the Simes (1986) procedure rejects $\cap_{i\in\{1,\dots,m\}}H_{0i}$ if and only if $|h_u(p,\alpha)| > 0$.

2. The Hochberg (1988) procedure uses $\tau_\alpha(i) = \alpha/(m+1-i)$.

3. The Rom (1990) procedure increases the power of the Hochberg (1988) procedure by replacing its critical values $\tau_\alpha(i) = \alpha/(m+1-i)$ by "sharper" values $\tau_\alpha(i) = c_i$. The $c_i$ are computed recursively as given in Rom (1990) and satisfy $c_i \nearrow$ for a fixed $\alpha$. Moreover, the $c_i$ are non-decreasing in $\alpha$.

4. The choice $\tau_\alpha(i) = i\alpha/m$ yields the Benjamini and Hochberg (1995) procedure.

5. The Benjamini and Yekutieli (2001) procedure controls the fdr under arbitrary dependence by applying the Benjamini and Hochberg (1995) procedure at the corrected constant threshold $\alpha/\left(\sum_{i=1}^m i^{-1}\right)$.

Similarly, the following step-down procedures satisfy Condition 3.11:

1. The Sidak (1967) correction uses $\tau_\alpha(i) = 1-(1-\alpha)^{1/(m+1-i)}$.

2. The choice $\tau_\alpha(i) = \alpha/(m+1-i)$ yields the Holm (1979) procedure.

3. The Shaffer (1986) procedure modifies the Holm (1979) procedure in order to obtain an increase in power. For the tests under consideration, let $0 \leq a_1 <$

$a_2 < \cdots < a_r \leq n$ be all possible numbers of true null hypotheses. Assuming that $H_{0(1)}, \ldots, H_{0(i-1)}$ are false, let $t_i = \max\{a_j : a_j \leq n - i + 1\}$ be the maximum possible number of true null hypotheses. The Shaffer (1986) procedure determines the minimal index $k$ such that $p_{(k)} > \alpha/t_k$ and then rejects $H_{0(1)}, \ldots, H_{0(k-1)}$. It can be obtained from a step-down procedure using $\tau_\alpha(i) = \alpha/t_i$, which is clearly continuous and non-decreasing in $\alpha$ for a fixed $i$. As $a_i \nearrow$ and thus $t_i \searrow$, $\tau_\alpha(i)$ is also non-decreasing in $i$ for a fixed $\alpha$.

For a given $\alpha^*$, by Lemma 3.12, all the procedures listed above are admissible for all but a null set of p-values $p^*$.

## 3.5 Using the framework in practice

The improved naive method derived in Example 3.2 is capable of computing test results which consist, up to a pre-specified error probability $\epsilon$, of sets of correctly rejected and correctly non-rejected hypotheses as well as of a set of undecided hypotheses. The following contains an example of such a classification.

Sandve et al. (2011) use their method `MCFDR` to classify a dataset of gene modifications (so-called H3K4me2-modifications) of Pekowska et al. (2010). This dataset consists of gene regions and gene modifications within each region, characterised by their midpoint. The beginning and the end of each region on the genome are normed to 0 and 1, respectively. The authors test if the gene modifications appear more often in a certain part of the gene region.

To be precise, Sandve et al. (2011) observe $k$ random points $Y_1, \ldots, Y_k$ in $[0, 1]$ (these are the midpoints of the gene modifications) and test the null hypothesis $H_0 : \mathbb{E}\left(\frac{1}{k}\sum_{i=1}^{k} Y_i\right) \geq 0.5$ against the alternative $H_1 : \mathbb{E}\left(\frac{1}{k}\sum_{i=1}^{k} Y_i\right) < 0.5$ using the test statistic $T = \frac{1}{k}\sum_{i=1}^{k} Y_i$. Each null hypothesis is tested by permuting the midpoints in each region while preserving their inter-point distances.

Sandve et al. (2011) first filter the dataset for genes with at least 10 modifications per gene region. Each such region becomes one hypothesis, leading to $m = 3465$ hypotheses (gene regions) under consideration. They evaluate the data using the procedure of Benjamini and Hochberg (1995) with a corrected testing threshold at level $0.1/\hat{\pi}_0(\hat{p})$, where $\hat{\pi}_0$ is the estimator of Pounds and Cheng (2006) introduced in Example 3.9 and $\hat{p}$ is an estimate of $p^*$ returned by `MCFDR`. Sandve et al. (2011) report 2747 significant hypotheses.

Nevertheless, the authors do not provide any guarantee on the correctness of their findings. Recomputing the results of Sandve et al. (2011) indeed shows considerable variability. To demonstrate this, we re-classify the H3K4me2 dataset using the `MCFDR` algorithm of Sandve et al. (2011) a total number of $r = 1000$ times. Let $p_i^s$ ($p_i^n$) be the empirical probability that hypothesis $H_{0i}$ is significant (non-significant) in these $r$ repetitions.

We are interested in measuring the randomness in the output of an algorithm and use $p_i^r = \min(p_i^s, p_i^n)$ as probability of $H_{0i}$ being randomly classified. We call all hypotheses having $p_i^r > 0.1$ "randomly classified" and denote their total number by $rc$. For `MCFDR` we observe that 195 hypotheses remain randomly classified on average.

We first use the (unimproved) naive method (as defined at the beginning of Example 3.2) with $s \in \{10^2, 10^3, 10^4\}$ samples per hypothesis to classify the same dataset. Table 3.1 shows the number of randomly classified hypotheses $rc$ observed for the naive method as a function of $s$ (second column). For $s = 10^2$, the total effort is comparable to the one of `MCFDR` and both methods yield equally high numbers of ran-

Table 3.1: Repeated application of the improved and the unimproved naive method to the same data.

| | naive method | improved naive method | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | with plug-in interval (Ex.3.9) | | | | with Hoeffding's interval (Ex.3.10) | | | |
| s | *rc* | rejected | non-rej. | undec. | *rc* | rejected | non-rej. | undec. | *rc* |
| $10^2$ | 196 | 0 | 161.8 | 3303.2 | 0 | 0 | 372.0 | 3093.0 | 0 |
| $10^3$ | 60 | 2386.0 | 487.5 | 591.5 | 0 | 2568.5 | 576.0 | 320.5 | 0 |
| $10^4$ | 18 | 2649.0 | 624.6 | 191.4 | 0 | 2697.3 | 661.7 | 106.0 | 0 |

$s$: number of samples drawn per hypotheses; $rc$: number of randomly classified hypotheses; rejected, non-rejected and undecided are average numbers based on 1000 repetitions.

dom decisions ($rc \approx 195$). For high precision ($s = 10^4$), up to 18 hypotheses remain inconsistently classified.

We then apply the improved naive method described in Example 3.2 to the same dataset using an overall error probability of $\epsilon = 0.01$. The improved method is stopped after having drawn $s$ samples per hypothesis. Table 3.1 shows rejected, non-rejected, undecided (see Remark 3.8) and randomly classified hypotheses. We evaluate both the plug-in interval for $\alpha^*$ introduced in Example 3.9 (columns three to six) as well as Hoeffding's confidence interval derived in Example 3.10 (columns seven to ten). For Hoeffding's interval, we use $\eta_n = \nu_n - \nu_{n-1}$ with $\nu_n = \frac{n}{n+s}\frac{\epsilon}{m+1}$, $n \in \mathbb{N}$.

Using a confidence interval for $\alpha^*$ based on Hoeffding's inequality (as opposed to the plug-in interval) yields considerably more decisions (rejections and non-rejections) and thus less undecided hypotheses for all ranges of precision.

Although for low numbers of samples many hypotheses remain undecided, the test results of the improved naive method are consistent in the sense that no hypothesis is randomly classified. The improved naive method therefore provides reliable test results and ensures repeatability. For a high precision ($s = 10^4$), the improved naive method with Hoeffding's interval for $\alpha^*$ yields around 2700 rejections and 660 non-rejections. The remaining 106 hypotheses are still unclassified, meaning that within this limited computational effort, no statement about these hypotheses (gene regions) should be made. The probability of the above results being correct is at least 0.99.

## 3.6 Discussion

We consider p-value based multiple testing under the assumption that the p-value of each hypothesis is not available and thus has to be approximated using Monte Carlo simulations. Although widely occurring in experimental studies, common methods for this scenario do not give any guarantee on how their test results relate to the one obtained if all p-values had been available.

Section 3.2 introduced a framework for Monte Carlo based multiple testing, both in terms of a general multiple testing procedure and a generic algorithm. Conditions on both the multiple testing procedure and the algorithm guarantee that the rejections and non-rejections returned by our generic algorithm are identical to the ones obtained with the p-values. A simplified condition for step-up and step-down multiple testing procedures is derived.

The framework thus extends the idea behind the `MMCTest` algorithm of Chapter 2 to a wider range of both Monte Carlo algorithms used to test multiple hypotheses as well as to a whole class of step-up and step-down procedures.

We demonstrate how to use our framework to modify established methods in such a way as to yield theoretical guarantees on their test results. As demonstrated on a class of commonly used methods, these modifications can easily be implemented in practice and come at virtually no additional computational cost.

Improved established methods, such as the improved naive method evaluated on a real data study, allow one to report multiple testing results as three sets: rejected, non-rejected and undecided hypotheses, together with an error bound on their correctness. We recommend any multiple testing result to be reported in this fashion.

# 4 QuickMMCTest – Higher accuracy for multiple testing corrections

## 4.1 Introduction

So far, Chapter 2 and Chapter 3 have focused on how to obtain theoretical guarantees on the correctness of decisions computed by algorithms implementing a multiple testing procedure. This chapter looks at multiple testing from a practical perspective: In order to evaluate scientific studies, it may be desirable to spend the (additional) effort needed to obtain a guarantee of correctness on certain hypotheses on a more precise ad-hoc classification instead, thus sacrificing the theoretical guarantee in order to minimise numbers of misclassifications. Such a complete classification of all hypotheses might be more useful than methods computing guaranteed decisions at the expense of leaving several hypotheses unclassified – especially as the total number of samples available in practice is finite and the runtime of methods such as `MMCTest` is high.

As before, we consider the scenario in which p-values cannot be computed exactly. They are approximated using Monte Carlo tests such as permutation tests or bootstrap tests. Permutation tests are widely used in practice as underlying models for biological phenomena are rarely known (Lourenco and Pires, 2014; Martínez-Camblor, 2014; Liu et al., 2013; Wu et al., 2013; Asomaning and Archer, 2012; Dazard and Rao, 2012).

As in Section 2.1 we are interested in the classification of all hypotheses obtained with the full knowledge of all p-values. In the case of permutation tests, for instance, these p-values are defined as the ones computed by exhaustively listing all permutations. For bootstrap tests the p-value is defined as the probability that a bootstrapped test statistic is at least as extreme as the observed test statistic.

A simple method to apply a multiplicity correction in the scenario under consideration is to first draw a constant number of samples for each hypothesis, then approximate its p-value using a conservative p-value estimate and finally use the estimates as input for the multiple testing procedure, thus treating them as if they were the p-values. This naive approach is widely used to evaluate multiple tests without knowledge of the p-values (Nusinow et al., 2012; Gusenleitner et al., 2012; Rahmatallah et al., 2012; Zhou et al., 2013; Li et al., 2012; Cohen et al., 2012).

However, the naive approach does not take into account that hypotheses whose p-values clearly lie in the rejection or non-rejection area of the multiple testing procedure should be allocated less samples than hypotheses whose p-values are closer to the threshold and thus harder to classify. This leaves considerable scope to improve upon the accuracy of the naive method.

This chapter introduces a sampling algorithm based on Thompson (1933) Sampling. Our approach, called `QuickMMCTest`, uses a Beta-binomial model on each p-value to adaptively decide which hypotheses need to receive more and which need less samples to obtain fairly clear decisions (rejections/ non-rejections) on all tests. One of its main features consists in avoiding to compute discrete p-value estimates at any stage. As highlighted in the chapter, `QuickMMCTest` thus circumvents imprecisions observed in methods using such discrete estimates. `QuickMMCTest` works with a variety of commonly used multiple testing procedures at constant testing thresholds and, moreover, with variable testing thresholds, that is thresholds which are functionals of the unknown p-values underlying the tests.

For the special case of the Bonferroni (1936) correction, empirical studies indicate that the allocation of samples computed by `QuickMMCTest` could be asympotically optimal in the sense that as the total number of samples drawn by `QuickMMCTest` goes to infinity, its allocation seems to mimic the optimal allocation minimising the expected number of misclassifications. This is demonstrated in Chapter 5.

The chapter is organised as follows. In Section 4.2 we introduce the set-up considered in this chapter, specifically the multiple testing procedure and the variable testing threshold used as well as the relation of our approach to Thompson (1933) Sampling. Section 4.2.1 presents `QuickMMCTest` and Section 4.2.2 discusses an alternative algorithm based on discrete p-value estimates, in particular its disadvantages in comparison to `QuickMMCTest`.

`QuickMMCTest` is evaluated in a simulation study in Section 4.3 for a variety of common multiple testing procedures (Bonferroni, 1936; Sidak, 1967; Holm, 1979; Simes, 1986; Hochberg, 1988; Benjamini and Hochberg, 1995; Benjamini and Yekutieli, 2001) using a constant testing threshold. The study shows that our approach yields a considerable decrease in the number of erroneously classified hypotheses in comparison to the naive approach on simulated data. In a second simulation study, we compare `QuickMMCTest` to a variety of commonly used methods (Besag and Clifford, 1991; Guo and Peddada, 2008; Sandve et al., 2011; Jiang and Salzman, 2012; Gandy and H., 2014) using the popular Bonferroni (1936) correction at a constant threshold, again confirming the high accuracy of the proposed approach.

We conclude with a discussion in Section 4.4.

Appendix C repeats the simulation studies conducted in Section 4.3 for a variable testing threshold.

`QuickMMCTest` has been implemented in an R-package (`simctest`, available on CRAN, The Comprehensive R Archive Network).

In the following, we will denote the density of the Beta distribution with shape parameters $\alpha$ and $\beta$ as $\text{Beta}(x; \alpha, \beta)$.

## 4.2 Two sampling algorithms based on Thompson Sampling

We assume that the p-values $p^* = (p_1^*, \dots, p_m^*)$ of the tests for $H_{01}, \dots, H_{0m}$ are not available analytically. Instead, we assume that it is possible to draw samples under the null hypothesis in order to approximate each p-value. To this end, we denote the total number of samples drawn for each of the $m$ hypotheses up to iteration $n$ by $k_n^i$ and the total number of exceedances observed among these $k_n^i$ samples by $S_n^i$, where $i \in \{1, \dots, m\}$. Moreover, the testing threshold $\alpha^*$ at which all hypotheses are tested can either be constant or a functional $\alpha(p^*)$ of the p-values $p^*$. In the latter case, $\alpha^*$ is itself unknown.

We are interested in computing $h(p^*, \alpha(p^*))$, where for a constant threshold,

$\alpha(p^*) = \alpha^* \in \mathbb{R}$ is independent of $p^*$ and known. Our aim is to approximate $h(p^*, \alpha(p^*))$ in such a way as to minimise the number of misclassifications, that is the number of decisions based on Monte Carlo sampling which differ from the ones obtained if the p-values $p^*$ had been available.

Our approach is based on an idea related to Thompson Sampling (Agrawal and Goyal, 2012; Thompson, 1933) and updates a Beta-Binomial model for each p-value in each iteration: Starting with a Beta$(1, 1)$ prior on each p-value, observing $S_n^i$ exceedances among $k_n^i$ samples results in a Beta$(1 + S_n^i, 1 + k_n^i - S_n^i)$ posterior.

The posterior distributions on the p-values are used to derive weights needed to allocate samples adaptively to all hypotheses. These weights can be computed in two different ways, leading to `QuickMMCTest` and an alternative approach presented in the following two subsections.

### 4.2.1 The `QuickMMCTest` algorithm

The idea of `QuickMMCTest` is to use the posterior distributions available in the Beta-Binomial model in the following way to obtain weights: In each iteration, all $m$ p-values are resampled from the posterior distributions. We then evaluate the multiple testing procedure on the $m$ resampled p-values and record the rejections and non-rejections. Repeating the above a fixed number of $R$ times allows one to compute an empirical probability that each hypothesis $H_{0i}$, $i \in \{1, \dots, m\}$, is rejected ($p_i^r$) and non-rejected ($1 - p_i^r$). The quantity $w_i = \min(p_i^r, 1 - p_i^r)$ can then be viewed as a measure of how stable the current decision (rejection, non-rejection) on $H_{0i}$ is.

In each iteration, a new batch of $\Delta$ samples is allocated to all $H_{0i}$ proportional to $w_i$, $i \in \{1, \dots, m\}$. This ensures that hypotheses already having a very stable decision only receive few new samples.

The above approach is summarised in Algorithm 4.1. Algorithm 4.1 has two parameters chosen by the user: the total number of samples $K$ the algorithm is allowed to spend and the number of iterations $n_{\max}$ (and thus the number of posterior updates). These two parameters determine the batch size $\Delta = K/n_{\max}$, that is the number of samples spent in each iteration. Alternative approaches in which $\Delta$ varies over time are possible. Section 4.3.3 shows that for more than roughly 10 iterations, the performance of Algorithm 4.1 does not substantially depend on the parameter $n_{\max}$ any more.

**Algorithm 4.1:** `QuickMMCTest`

```
   input: $K$, $n_{\max}$
1  $\Delta \leftarrow \lfloor K/n_{\max} \rfloor$; $k_0^i \leftarrow 0$; $S_0^i \leftarrow 0$, $w_i \leftarrow 1$, $i \in \{1, \dots, m\}$;
2  for $n \leftarrow 1$ to $n_{\max}$ do
3      if $n > 1$ then
4          $r_i \leftarrow 0$, $i \in \{1, \dots, m\}$;
5          repeat $R$ times
6              $p_i \sim \text{Beta}(1 + S_{n-1}^i, 1 + k_{n-1}^i - S_{n-1}^i)$ indep., $i \in \{1, \dots, m\}$;
7              $r_i \leftarrow r_i + \mathbb{I}(i \in h(p, \alpha(p)))$, where $p = (p_1, \dots, p_m)$, $i \in \{1, \dots, m\}$;
8          $w_i \leftarrow \min(r_i/R, 1 - r_i/R)$, $i \in \{1, \dots, m\}$;
9          if $\sum_{j=1}^m w_j = 0$ then
10             $w_i \leftarrow 1$, $i \in \{1, \dots, m\}$;
11     $(w_1, \dots, w_m) \leftarrow (w_1, \dots, w_m) / \left( \sum_{j=1}^m w_j \right)$;
12     Draw $\Delta$ samples with weights $(w_1, \dots, w_m)$ using residual sampling and
       update $k_n^i$, $S_n^i$, $i \in \{1, \dots, m\}$;
13 return $(S_n^1, \dots, S_n^m)$, $(k_n^1, \dots, k_n^m)$;
```

Algorithm 4.1 uses residual sampling (Liu and Chen, 1998) in line 12: To guarantee a deterministic minimal allocation of samples to each hypothesis, we first draw $\lfloor w_i \Delta \rfloor$ samples for each $H_{0i}$, $i \in \{1, \dots, m\}$. The remaining $\Delta - \sum_{j=1}^m \lfloor w_j \Delta \rfloor$ samples are then allocated to all hypotheses by drawing one sample at a time with weights proportional to $(w_1\Delta - \lfloor w_1\Delta\rfloor, \dots, w_m\Delta - \lfloor w_m\Delta\rfloor)$. For each hypothesis $H_{0i}$, $i \in \{1, \dots, m\}$, both $k_n^i$ and $S_n^i$ are updated according to the total number of samples drawn in this fashion and the number of exceedances observed among these samples.

Alternatively, one could replace the residual sampling by simple multinomial sampling or other methods used in, for instance, particle filters.

Determining the weights is computationally fast as it only requires $R$ draws per hypothesis from a Beta distribution as opposed to drawing samples from the data (for instance via permutations which can be costly). The dependence of Algorithm 4.1 on $R$ investigated in Section 4.3.4 shows that although higher values of $R$ result in a more precise allocation of samples, increasing $R$ beyond roughly $R = 1000$ does not considerably improve performance.

Decisions on all hypotheses can be obtained with Algorithm 4.1 in various ways. Naively, one could compute $h(\hat{p}, \alpha(\hat{p}))$, where $\hat{p} = (\hat{p}_1, \dots, \hat{p}_m)$ is a vector of estimates $\hat{p}_i = (S_{n_{\max}}^i + 1)/(k_{n_{\max}}^i + 1)$, $i \in \{1, \dots, m\}$, from the last iteration of the algorithm computed using a pseudo-count (Davison and Hinkley, 1997) in both the numerator and the denominator.

In the entire chapter, we do not consider computing unbiased p-value estimates without a pseudo-count (that is $S_{n_{\max}}^i / k_{n_{\max}}^i$) in order to classify all hypotheses: such estimates lead to tests not keeping the prescribed error level (Davison and Hinkley, 1997; Manly, 1997; Edgington and Onghena, 1997).

A more sophisticated approach than the one based on p-value estimates is to repeat the classification all hypotheses using draws from the Beta distributions in lines 4 to 7 of Algorithm 4.1. Each hypothesis $H_{0i}$, $i \in \{1, \dots, m\}$, is then rejected if and only if $r_i/R > 0.5$, that is if $H_{0i}$ was predominantly rejected based on resampled p-values. We recommend to obtain decisions in this fashion as this approach turns out to empirically result in less misclassifications than approaches based on discrete p-value estimates computed with a pseudo-count (Section 4.3.5). The cutoff of 0.5 is arbitrary and can be replaced by higher (lower) values to make Algorithm 4.1 more (less) conservative.

Algorithm 4.1 is evaluated in a simulation study in the next section, where we will use the latter approach with a cutoff of 0.5 to obtain decisions on all hypotheses.

### 4.2.2 An alternative approach using discrete p-value estimates

Computing the weights in Algorithm 4.1 using samples from the Beta distributions of all p-values introduces an additional source of randomness into the algorithm. To avoid this and to be computationally more efficient, weights can also be computed using the cumulative distribution function (cdf) of the Beta distribution – given that an efficient implemention of the cdf of the Beta distribution is available. This modification is discussed in the present section and comes at the cost of computing p-value estimates which, as shown at the end of this section, also has severe disadvantages.

We use the posterior distribution of each $p_i^*$ to calculate the two probabilities that $p_i^*$ lies below and above the estimated cutoff $\hat{\tau}$ separating the rejection from the non-rejection area. Similarly to Algorithm 4.1, we then allocate the next batch of $\Delta$ samples to each hypothesis $H_{0i}$, $i \in \{1, \ldots, m\}$, according to the weighted minimal probability of the two. This ensures that hypotheses having a considerable proportion of the posterior mass on either side of the threshold (and thus a fairly clear decision on whether their p-values lie in the rejection or non-rejection area) only receive few new samples.

We approximate $\hat{\tau}$ as the midpoint of the last rejected and first non-rejected p-value estimate, that is $\hat{\tau} = (\max\{\hat{p}_i : i \in h(\hat{p}, \alpha(\hat{p}))\} + \min\{\hat{p}_i : i \notin h(\hat{p}, \alpha(\hat{p}))\})/2$, where $\hat{p} = (S_n^1/k_n^1, \ldots, S_n^m/k_n^m)$ is a vector of p-value estimates in iteration $n$ and $\alpha(\hat{p})$ is the plug-in estimate of the threshold using the current estimates. As a convention, we define $\max\emptyset := 0$ and $\min\emptyset := 1$.

The missing pseudo-count in the definition of $\hat{\tau}$ is crucial: Using a pseudo-count causes all p-value estimates $\hat{p}_i$ to be bounded below by $1/(k_n^i + 1)$, where $k_n^i$ is the number of samples taken for hypothesis $H_{0i}$, $i \in \{1, \ldots, m\}$, up to iteration $n$. Due to the multiplicity correction, this lower bound can be larger than the testing threshold itself, in which case all hypotheses are consistently classified as non-rejected in each repetition, thus producing meaningless results and an estimated cutoff equal to the largest p-value estimate in every iteration. Section C.1 contains further details. It it thus necessary to compute $\hat{\tau}$ without a pseudo-count, even at the expense of over-rejecting hypotheses.

The above approach is summarised in Algorithm 4.2 which, apart from the new computation of the weights replacing lines 4 to 8 in Algorithm 4.1, works identically to `QuickMMCTest`. We therefore only state all modified lines in Algorithm 4.2.

**Algorithm 4.2:** Alternative computation of weights replacing lines 4 to 8 in Algorithm 4.1

```
1 $\hat{p} \leftarrow (S_n^1/k_n^1, \ldots, S_n^m/k_n^m)$;
2 $\hat{\tau} \leftarrow (\max\{\hat{p}_i : i \in h(\hat{p}, \alpha(\hat{p}))\} + \min\{\hat{p}_i : i \notin h(\hat{p}, \alpha(\hat{p}))\})/2$;
3 $\hat{w}_i \leftarrow \int_0^{\hat{\tau}} \mathrm{Beta}(x; 1 + S_n^i, 1 + k_n^i - S_n^i)dx,\ i \in \{1, \ldots, m\}$;
4 $w_i \leftarrow \min(\hat{w}_i, 1 - \hat{w}_i),\ i \in \{1, \ldots, m\}$;
```

A final testing result of Algorithm 4.2 can be computed in the same fashion as done for Algorithm 4.1 by either classifying all hypotheses using p-value estimates computed with a pseudo-count or by computing empirical rejection probabilities (see Section 4.2.1). Section 4.3.5 contains a study comparing the accuracy of both Algorithm 4.1 and 4.2 for the two decision techniques.

This study in Section 4.3.5 shows that Algorithm 4.2 indeed suffers from not being able to record any rejections when using a pseudo-count to compute p-value estimates after termination, and moreover that Algorithm 4.2 yields up to twice as many erroneously rejected hypotheses (false findings) as Algorithm 4.1. Therefore, the following main simulation study in Section 4.3 focuses on `QuickMMCTest` instead of Algorithm 4.2 when comparing our approach to published methods.

## 4.3 Simulation study

We evaluate `QuickMMCTest` on a simulated dataset in two ways. First, we compare the performance of `QuickMMCTest` to the one of a naive sampling method using a variety of commonly used multiple testing procedures at a constant testing threshold (Section 4.3.1). Second, we fix the Bonferroni (1936) procedure as multiple testing procedure and compare `QuickMMCTest` to a variety of common methods published in the literature (Section 4.3.2).

Section 4.3 also investigates the influence of the number of updates $n_{\max}$ on `QuickMMCTest` in Section 4.3.3 as well as the dependence of `QuickMMCTest` on the parameter $R$ controlling the accuracy used to compute weights (Section 4.3.4).

Moreover, both Algorithm 4.1 and Algorithm 4.2 are compared using two different approaches to report final testing results: one based on p-values and one based on empirical rejection probabilities (Section 4.3.5).

As a model for p-value distributions occuring in real data studies, we use the model of Sandve et al. (2011) consisting of a proportion $\pi_0 \in [0,1]$ (the proportion of true null hypotheses) drawn from a Uniform$[0,1]$ distribution and the remaining proportion $1-\pi_0$ drawn from a Beta$(0.25, 25)$ distribution. A high proportion $\pi_0$ is to be expected in realistic scenarios. We therefore use $\pi_0 = 0.9$ and draw $m = 5000$ p-values from the above mixture distribution. These p-values remain fixed throughout the remainder of this chapter. Comparing the test result returned by each algorithm under investigation to the one obtained by applying the multiple testing procedure to the fixed set of $m$ p-values allows one to compute numbers of misclassifications. By a misclassification we refer to any decision (rejection, non-rejection) of an individual hypothesis which is different from the one obtained by applying the multiple testing procedure to the fixed p-values.

In the entire section, we use a total number of $n_{\max} = 10$ iterations (posterior updates) for `QuickMMCTest` and always estimate empirical rejection and non-rejection probabilities using $R = 1000$ draws from the Beta posteriors.

All results are based on 1000 repetitions.

### 4.3.1 Comparison to a naive method using various multiple testing procedures

We compare `QuickMMCTest` to the naive approach introduced in Section 4.1 on a variety of commonly used multiple testing procedures, precisely the step-up procedures of Bonferroni (1936), Simes (1986), Hochberg (1988), Benjamini and Hochberg (1995) and Benjamini and Yekutieli (2001), as well as to the step-down procedures of Sidak (1967) and Holm (1979).

The naive method uses a fixed number of $s$ samples to estimate each p-value

Table 4.1: Average misclassification numbers (average numbers of erroneously rejected hypotheses in brackets) for the naive method at a low effort ($s = 1000$) and a high effort ($s = 10000$) compared to `QuickMMCTest` (Alg. 4.1) for common multiple testing procedures at the constant testing threshold 0.1.

| | low effort ($s$=1000) | | high effort ($s$=10000) | |
|---|---|---|---|---|
| procedure | naive | Alg. 4.1 | naive | Alg. 4.1 |
| Bonferroni (1936) | 87 (0) | 43.8 (2.6) | 87 (0) | 3 (1.7) |
| Simes (1986) | 32 (9.6) | 2 (0.9) | 9 (3.8) | 0.1 (0.1) |
| Hochberg (1988) | 87 (0) | 43.4 (2.5) | 87 (0) | 3.2 (2) |
| Benjamini and Hochberg (1995) | 31.9 (9.5) | 2 (1) | 9.1 (3.8) | 0.1 (0.1) |
| Benjamini and Yekutieli (2001) | 162 (0) | 14.5 (3.3) | 22 (5.5) | 0.6 (0.6) |
| Sidak (1967) | 90 (0) | 36.3 (2.9) | 90 (0) | 3.5 (1.6) |
| Holm (1979) | 88 (0) | 39.5 (3.2) | 88 (0) | 3.4 (2.1) |

and then applies the multiple testing procedure to the estimates, thus treating the estimates as if they were the p-values. Each p-value $p_i^*$ is estimated using a pseudo-count (Davison and Hinkley, 1997) as $\hat{p}_i = (e_i + 1)/(s + 1)$, where $e_i$ is the number of exceedances observed for $H_{0i}$, $i \in \{1, \dots, m\}$, and $s$ is the number of samples. This naive approach is widely used to evaluate multiple tests without knowledge of the analytical p-values (Nusinow et al., 2012; Gusenleitner et al., 2012; Rahmatallah et al., 2012; Zhou et al., 2013; Li et al., 2012; Cohen et al., 2012).

We assume that the testing threshold is fixed at $\alpha^* = 0.1$ and evaluate the naive method as well as `QuickMMCTest` on the fixed p-values by calculating numbers of misclassifications. For this, the naive method is repeatedly applied at both a low effort (using a constant number of $s = 1000$ samples to estimate the p-value of each hypothesis) and a high effort ($s = 10000$) and in both cases `QuickMMCTest` is applied with a matched effort. Alternatively, one could also consider comparing several methods by differentiating which hypotheses generated from the null and the alternative are erroneously classified.

Table 4.1 presents simulation results. It shows that the phenomenon of consistently non-rejecting all hypotheses due to the usage of a pseudo-count can be observed in the naive method. As described in Section 4.2.2 and Section C.1, the pseudo-count causes all p-value estimates to be bounded from below and thus may result in all hypotheses being consistently non-rejected. In this case, the number of misclassifications is hence equal to the number of the undetected rejections. This phenomenon occurs when applying the naive method to the procedures of Bonferroni (1936), Hochberg (1988), Sidak (1967) and Holm (1979), both at a low and at a high effort. For the Benjamini and Yekutieli (2001) procedure, results are meaningful only at a high effort. This can be seen in Table 4.1 by looking at the number of erroneously rejected hypotheses (given in brackets behind the misclassification numbers), which is zero (indicating that a method either perfectly classifies rejections or that no rejections are recorded at all, where the latter applies).

The naive method is only able to compute meaningful results at low effort for the two procedures of Simes (1986) and Benjamini and Hochberg (1995), though results still contain around 30 misclassifications on average. Most importantly, the naive method erroneously rejects considerably more hypotheses than `QuickMMCTest` (in the cases where rejections can be observed at all), thus reporting more false findings which are undesired in practice.

In contrast to the naive method, `QuickMMCTest` does not rely on computing p-value estimates and therefore computes meaningful results for all procedures. These are very accurate for the procedures of Simes (1986) and Benjamini and Hochberg (1995) at low effort. For all other methods, `QuickMMCTest` yields around 35 to 45 misclassifications.

When applying the naive method at a high effort, drawing $s = 10000$ samples per hypothesis is still not enough to observe any rejections for the procedures of Bonferroni (1936), Hochberg (1988), Sidak (1967) and Holm (1979). For the procedures of Simes (1986), Benjamini and Hochberg (1995) and Benjamini and Yekutieli (2001), the naive method yields around 10 to 20 misclassifications.

The advantages of the more efficient allocation scheme of `QuickMMCTest` becomes even more apparent at a high effort. When being allowed to use more samples, `QuickMMCTest` yields considerably less misclassifications than the naive method for all testing procedures considered in this section and essentially no misclassifications for the procedures of Simes (1986) and Benjamini and Hochberg (1995). Similarly to the comparison at a low effort, `QuickMMCTest` again erroneously rejects considerably less hypotheses than the naive method, a feature desired for practical use.

The above results are confirmed by a second study with a variable testing threshold which depends on the unknown p-values: to be precise, we correct the testing threshold using $\alpha(p^*) = \alpha^*/\hat{\pi}_0(p^*)$, where $\alpha^* = 0.1$ and $\hat{\pi}_0(p) = \min\left(1, 2/m \sum_{i=1}^{m} p_i\right)$ is a robust estimate of the proportion $\pi_0$ of true null hypotheses of Pounds and Cheng (2006). Results for this variable testing threshold are contained in Section C.2 and are qualitatively similar to the ones in Table 4.1.

### 4.3.2 Comparison to a variety of common methods

A second simulation study compares `QuickMMCTest` to common algorithms available in the literature to test multiple hypotheses based on Monte Carlo sampling. These algorithms are the naive method and the algorithms of Besag and Clifford (1991), Guo and Peddada (2008), Sandve et al. (2011), Jiang and Salzman (2012) as well as Gandy and H. (2014).

The naive method is used as a reference for setting the effort: we define low effort as $K = 1000m$, that is the effort equivalent to spending $s = 1000$ samples per hypothesis, and similarly high effort as $K = 10000m$.

All methods are run with standard parameters suggested by their authors:

1. The naive method was run with $s = 1000$ samples per hypothesis at low effort and $s = 10000$ samples at a high effort. Estimates were computed using a pseudo-count (Davison and Hinkley, 1997) as $(e + 1)/(s + 1)$, where $e$ is the number of exceedances and $s$ is the number of samples. A decision on all hypotheses is obtained by evaluating the multiple testing procedure on the estimates.

2. In order to match the overall effort, the algorithm of Besag and Clifford (1991) was run by repeatedly drawing one sample for each hypothesis until either $h = 20$ exceedances were observed (as proposed by the authors) or the total effort was reached. P-values were computed with a pseudo-count in the numerator as proposed by the authors.

3. The algorithm of Guo and Peddada (2008) was implemented using a geometric sequence $B_0 \leq B_1 \leq \ldots B_N$, where $N = 9$, $B_0 = 10$, $B_{i+1} = aB_i$ and the geometric increase $a$ was computed in order to match the overall effort. Confidence intervals were computed using the method of Clopper and Pearson (1934) as proposed by the authors. P-value estimates were computed with a pseudo-count in both the numerator and denominator.

4. The `MCFDR` algorithm of Sandve et al. (2011) is the only one whose effort cannot easily be matched to a pre-specified total effort. We therefore tune the only parameter $h$ of `MCFDR` to meet the upper bound for the effort: for $h = 65$, the effort of `MCFDR` roughly equals $K = 1000m$, and for $h = 650$, the effort roughly equals $K = 10000m$, where $m = 5000$ is the number of hypotheses. In both cases, `MCFDR` is stopped on reaching $K$ samples in order to have a fair comparison, and both Algorithm 4.1 and 4.2 are applied after `MCFDR` using at most the number of samples `MCFDR` spent instead of the actual $K = 1000m$ or $K = 10000m$. P-value estimates were computed with a pseudo-count in both the numerator and denominator.

5. The method of Jiang and Salzman (2012) was run with parameters $a = 10$ and $\delta = 0.01$ as proposed by the authors in the simulation study included in their article. One new sample is drawn per hypothesis in each iteration and p-values were computed as proposed in the original article.

6. The `MMCTest` algorithm was run as suggested in Chapter 2. We used 10 iterations and a geometric increase $a$ in `MMCTest` set to the same value as the one used in the implemention of the algorithm of Guo and Peddada (2008) in order to exactly match the overall effort. P-value estimates were computed with a pseudo-count in both the numerator and denominator.

We apply all methods to the $m = 5000$ p-values fixed in Section 4.3 using the popular Bonferroni (1936) correction at a constant threshold of 0.1.

Table 4.2 shows the simulation results for both a low and a high effort. For a low effort, Table 4.2 demonstrates that due to the very low threshold of $0.1/m$

Table 4.2: Average misclassification numbers (average numbers of erroneously rejected hypotheses in brackets) for common methods compared to `QuickMMCTest` (Algorithm 4.1) for the Bonferroni (1936) correction at the constant threshold 0.1.

| | low effort ($K = 1000m$) | high effort ($K = 10000m$) |
|---|---|---|
| Naive method | 87 (0) | 87 (0) |
| Besag and Clifford (1991) | 87 (0) | 4.9 (2.4) |
| Guo and Peddada (2008) | 87 (0) | 4.5 (2.1) |
| Sandve et al. (2011) | 87 (0) | 19 (1.6) |
| Jiang and Salzman (2012) | 87 (0) | 16.3 (3.7) |
| Gandy and H. (2014) | 87 (0) | 5 (2.2) |
| `QuickMMCTest` | 43.7 (2.6) | 3 (1.7) |

$K$: total number of samples; $m$: number of hypotheses.

for Bonferroni (1936), all methods except for `QuickMMCTest` are unable to compute p-value estimates with a resolution sufficient to detect any rejections. They thus non-reject all hypotheses, leading to misclassification numbers equal to the 87 rejections observed when applying the Bonferroni (1936) correction to the fixed p-values. As already seen in Section 4.3.1, `QuickMMCTest` on the other hand does not suffer from this problem.

For a high effort, most methods compute acceptable test results with around 5 misclassifications with the exception of the naive method which is still unable to detect any rejections.

Table 4.3 repeats the previous comparison using the Benjamini and Hochberg (1995) procedure controlling the false discovery rate. Due to the less conservative nature of the Benjamini and Hochberg (1995) procedure, all methods are able to compute meaningful test results at both a low and a high effort. The naive method and the one of Besag and Clifford (1991) perform poorly in this new scenario. The

Table 4.3: Average misclassification numbers (average numbers of erroneously rejected hypotheses in brackets) for common methods compared to `QuickMMCTest` using the Benjamini and Hochberg (1995) procedure. Constant threshold 0.1.

| | low effort | high effort |
|---|---|---|
| | ($K = 1000m$) | ($K = 10000m$) |
| Naive method | 31.9 (9.7) | 9.1 (3.7) |
| Besag and Clifford (1991) | 18.3 (7.5) | 18.3 (7.4) |
| Guo and Peddada (2008) | 4.5 (2.1) | 0.3 (0.3) |
| Sandve et al. (2011) | 10 (4) | 2.8 (1.3) |
| Jiang and Salzman (2012) | 13.2 (4.9) | 3.5 (1.6) |
| Gandy and H. (2014) | 9.9 (4.1) | 0.8 (0.6) |
| `QuickMMCTest` | 2 (1) | 0.1 (0.1) |

$K$: total number of samples; $m$: number of hypotheses.

method of Guo and Peddada (2008) performs very well and is only outperformed by `QuickMMCTest` at a low effort (yielding half as many misclassifications as Guo and Peddada (2008) and a multiple fold decrease compared to all other methods). At a high effort, Guo and Peddada (2008) perform comparably to `QuickMMCTest`.

The similar performance of the algorithms of Guo and Peddada (2008), Gandy and H. (2014) as well as `QuickMMCTest` is not a coincidence: Both Guo and Peddada (2008) as well as Gandy and H. (2014) use confidence intervals for all p-values in connection with a monotonicity property of the multiple testing procedure of Tamhane and Liu (2008) to stop the sampling for certain hypotheses, thus concentrating samples on hypotheses which are hard to classify. Nevertheless, neither Guo and Peddada (2008) nor Gandy and H. (2014) use any weights to tune the allocation of samples to individual hypotheses, a feature attempted in `QuickMMCTest` which yields another improvement in accuracy compared to the other two methods.

The above results are consistent even for a variable testing threshold: A second

Table 4.4: Number of updates $n_{\max}$ against misclassifications for `QuickMMCTest`.

| $n_{\max}$ | 1 | 2 | 5 | 10 | 20 | 50 | 100 | 200 |
|---|---|---|---|---|---|---|---|---|
| misclassifications | 31.735 | 10.594 | 3.448 | 1.967 | 1.689 | 1.628 | 1.571 | 1.572 |

simulation study contained in Section C.3 confirms the above results when comparing `QuickMMCTest` to the same set of algorithms using the variable testing threshold of Pounds and Cheng (2006).

### 4.3.3 Dependence on the number of updates

How does the performance of `QuickMMCTest` (Algorithm 4.1) depend on the parameter $n_{\max}$ controlling the number of iterations and thus the number of posterior updates?

`QuickMMCTest` was run on the fixed p-values (Section 4.3) using $R = 1000$ samples from the Beta posteriors, the Benjamini and Hochberg (1995) procedure and a total effort of $K = 1000m$, where $m = 5000$ is the number of hypotheses. The parameter $n_{\max}$ was varied.

Table 4.4 shows simulation results. As expected, the number of misclassifications decreases first with an increasing number of updates due to the more accurate computation of weights. This effect dominates until, for even larger values of $n_{\max}$ than displayed in Table 4.4, the available total number of samples for all hypotheses per iteration is so low that hypotheses with very low weights essentially do not receive any samples any more per iteration, thus increasing again their susceptibility to misclassifications. Running `QuickMMCTest` with more updates also considerably increases its runtime, therefore using $n_{\max} = 10$ or $n_{\max} = 20$ seems reasonable as it seems to yield a good trade-off between speed and accuracy. The choice $n_{\max} = 10$ is used as

Table 4.5: Parameter $R$ against misclassification numbers for `QuickMMCTest`.

| $R$ | 10 | 100 | 1000 | 10000 |
|---|---|---|---|---|
| misclassifications | 4.629 | 2.314 | 2.052 | 2.022 |

a default choice in Section 4.3.

### 4.3.4 Dependence on the number of repetitions to estimate rejection probabilities

Next, the influence of the parameter $R$ on `QuickMMCTest` is evaluated using the same set-up as in Section 4.3.3. The parameter $R$ controls the accuracy with which weights are estimated in each iteration. The default choice of $n_{\max} = 10$ determined in Section 4.3.3 is kept fixed.

Results are shown in Table 4.5. Not surprisingly, the test results returned by Algorithm 4.1 become more and more accurate with an increasing effort of the weights. In order to not spend too much time on the computation of weights, using $R = 1000$ as a default choice (see Section 4.3) seems to yield a reasonable trade-off between speed and accuracy.

### 4.3.5 Reporting testing results in two different ways

Two methods to compute a final testing result with `QuickMMCTest` (Algorithm 4.1) and Algorithm 4.2 are considered: classifications based on p-value estimates computed using a pseudo-count $\hat{p}_i = (S^i_{n_{\max}} + 1)/(k^i_{n_{\max}} + 1)$, where $S^i_n$, $k^i_n$ and $n_{\max}$ are as in Algorithm 4.1, as well as classifications based on empirical rejection probabilities $r_i/R$

Table 4.6: Reporting testing results in two different ways. Average misclassification numbers (average numbers of erroneously rejected hypotheses in brackets) at a low effort ($K = 1000m$) for Algorithm 4.1 (`QuickMMCTest`) and Algorithm 4.2 for common multiple testing procedures at the constant testing threshold 0.1.

| | Algorithm 4.1 | | Algorithm 4.2 | |
|---|---|---|---|---|
| procedure | p.c. | e.r.p. | p.c. | e.r.p. |
| Bonferroni (1936) | 64.5 (1) | 43.8 (2.6) | 87 (0) | 26.3 (5.5) |
| Simes (1986) | 2 (0.9) | 2 (0.9) | 2.1 (1) | 2.1 (1) |
| Hochberg (1988) | 64.2 (1) | 43.4 (2.5) | 87 (0) | 22.7 (5.8) |
| Benjamini and Hochberg (1995) | 2 (1) | 2 (1) | 2 (1) | 2 (1) |
| Benjamini and Yekutieli (2001) | 15 (2.8) | 14.5 (3.3) | 9.8 (3.3) | 9.7 (3.4) |
| Sidak (1967) | 74.4 (0.6) | 36.3 (2.9) | 89.5 (0) | 24.4 (5) |
| Holm (1979) | 76.5 (0.5) | 39.5 (3.2) | 87.5 (0.1) | 45.5 (3.8) |

p.c.: pseudo-count; e.r.p.: empirical rejection probabilities.

(compared to a cutoff of 0.5, see Section 4.2.1).

We repeat the simulation study presented in Section 4.3.1 for a low effort using the fixed p-value distribution of Section 4.3. In particular, the results reported for Algorithm 4.1 using empirical rejection probabilities correspond to the ones reported in Table 4.1.

Table 4.6 shows simulation results. For Algorithm 4.1, using empirical rejection probabilities instead of pseudo-count estimates to classify hypotheses yields, on average, a considerable decrease in the number of misclassifications.

Occasionally, testing results of Algorithm 4.2 obtained with pseudo-count estimates contain no rejections (for the procedures of Bonferroni (1936) and Hochberg (1988)) and are thus not suitable for practical use. Although Algorithm 4.2 yields less misclassifications than `QuickMMCTest` for some procedures, Algorithm 4.2 still relies on p-value estimates and is thus susceptible to incurring zero rejections at other efforts or other datasets. We therefore do not recommend using Algorithm 4.2 with pseudo-count estimates. On the other hand, results of Algorithm 4.2 obtained with empirical rejection probabilities contain up to twice as many erroneously rejected hypotheses (and thus false findings) than the ones of Algorithm 4.1. We thus recommend to use `QuickMMCTest`.

At a high effort, both Algorithm 4.1 and 4.2 yield equally precise results for both p-value estimates and empirical rejection probabilities (table not shown).

## 4.4 Discussion

Assuming the main scenario underlying this thesis in which it is not possible to compute p-values analytically for all tests, we aim to use Monte Carlo samples to approximate $h(p^*, \alpha(p^*))$ as accurately as possible.

An idea based on Thompson (1933) Sampling is proposed to efficiently allocate samples to multiple hypotheses. We derived an iterative algorithm based on this principle, called `QuickMMCTest`, which adaptively allocates more samples to hypotheses whose decision (rejected/ non-rejected) is still unstable at the expense of allocating less samples to hypotheses which can easily be classified.

`QuickMMCTest` has two main features: First, it never computes any p-value estimates during its run, thus avoiding to incur consistently non-rejecting all hypotheses as observed in other methods published in the literature. Second, computing final testing results based on empirical rejection probabilities (that is empirical probabilities of being rejected or non-rejected obtained through repeated classification of resampled p-values) instead of p-value estimates empirically results in less erroneously classified hypotheses. The algorithm works for a variety of common multiple testing procedures at both a constant as well as a variable testing threshold.

`QuickMMCTest` was evaluated in a simulation study. By comparing its performance to both a widely used naive sampling method for a variety of commonly used multiple testing procedures as well as to a variety of algorithms published in the literature, we demonstrated that `QuickMMCTest` yields meaningful testing results even at a low effort and up to a multiple fold decrease in the number of erroneously classified hypotheses at a high effort.

# 5 Optimal allocation of samples

## 5.1 Introduction

The final chapter picks up the discussion of the `QuickMMCTest` algorithm introduced in Chapter 4. This algorithm has proved to work excellently in connection with a number of multiple testing procedures, thus raising the natural question if the allocation of samples returned by `QuickMMCTest` is optimal in a certain sense.

This chapter contains preliminary work. All results presented in this chapter are not yet proven and rely on simulations.

We are interested in deriving the optimal allocation of a finite number of samples to a finite number of hypotheses tested using the Bonferroni (1936) correction. By optimal we refer to the allocation which minimises the total expected number of misclassified hypotheses, that is the expected number of hypotheses whose decisions (rejected, non-rejected) differ from the ones obtained with the p-values. Recent examples for studies evaluated with the Bonferroni (1936) correction and approximated p-values, even though the optimal allocation of samples was not guaranteed in these studies, include Thulin (2014); Zhang (2008); Kim et al. (2007).

Several methods to compute significant and non-significant hypotheses based on approximated p-values are available in the literature. For instance the method of Besag and Clifford (1991), the approaches by Lin (2005), van Wieringen et al. (2008), Guo and Peddada (2008), the `MCFDR` algorithm of Sandve et al. (2011) or the `MMCTest` algorithm of Gandy and H. (2014). However, it is unclear how the allocation of samples to each hypothesis computed by any of the algorithms aforementioned compares to the optimal allocation.

In Section 5.2 we will first state a mathematical formulation of the problem under investigation (Section 5.2.1). We then solve for the optimal allocation minimising the overall expected number of misclassifications under the assumption that the number of samples to be allocated can take real values and can be modelled using a normal approximation (Sections 5.2.2 and 5.2.3). This is achieved by solving a suitable optimisation problem using a Kuhn and Tucker (1951) constraint.

Section 5.3 indicates empirically that `QuickMMCTest` introduced in Chapter 4 might be close to asympotically imitating the optimal allocation of samples.

As in practice any allocation of samples is discrete, the original discrete optimisation problem (without normal approximation) is heuristically solved with a greedy algorithm to indicate that both the real-valued (normal approximated) and the integer optimal solutions might not differ by much (Section 5.4).

The chapter concludes with a discussion in Section 5.5.

## 5.2 Deriving the optimal allocation for a normal approximation

The following section introduces the problem under investigation in this chapter and presents a mathematical formulation as an optimisation problem. The optimal allocation minimising the expected number of misclassifications is derived with the help of a Kuhn and Tucker (1951) constraint.

### 5.2.1 Formulation of the problem

Suppose we are interested in testing $m$ hypotheses $H_{01}, \dots, H_{0m}$ for statistical significance using the Bonferroni (1936) correction at a fixed threshold $\alpha \in (0,1)$. As usual, the unknown p-value underlying each hypothesis $H_{0i}$ is denoted by $p_i^*$, $i \in \{1, \dots, m\}$.

In all sections, the multiple testing procedure $h$ is assumed to be the Bonferroni (1936) correction returning the indices of rejected hypotheses, defined as $h(p, \alpha) = \{i : p_i \leq \alpha\}$, where $p = (p_1, \dots, p_m)$ is a vector of $m$ p-values. Typically, the threshold $\alpha$ is a fixed significance level $\alpha^*$ divided by the number of hypotheses $m$ to correct for multiple tests, that is $\alpha = \alpha^*/m$.

As the p-values $p_i^*$ are unknown, we assume that Monte Carlo methods are used to approximate them as $\hat{p}_i = S_i/k_i$ without a pseudo-count, where $S_i$ is the number of exceedances observed among $k_i$ samples drawn for $H_{0i}$. The exceedances can be modelled as $S_i \sim \text{Binomial}(k_i, p_i^*)$.

We assume that in practical applications in which p-values are unknown, $H_{0i}$ is rejected if $S_i/k_i \leq \alpha$, that is if its p-value estimate $\hat{p}_i$ is below the constant threshold $\alpha$. All remaining hypotheses are non-rejected. This approach to computing rejections and non-rejections is employed in all the real data studies mentioned in Section 5.1.

We are interested in finding the allocation of samples $k_1, \dots, k_m$ to the hypotheses $H_{01}, \dots, H_{0m}$ which minimises the expected number of misclassifications, subject to the constraint that $\sum_{i=1}^m k_i = K$ for a total number of samples $K \in \mathbb{N}$ specified in advance by the user.

Let

$$M_i = \{S_i/k_i > \alpha \wedge p_i^* \leq \alpha\} \cup \{S_i/k_i \leq \alpha \wedge p_i^* > \alpha\}$$

be the event that the hypothesis $H_{0i}$ is misclassified. For the Bonferroni (1936) correction under consideration, this event occurs if the p-value estimate $\hat{p}_i = S_i/k_i$ and the p-value $p_i^*$ for $H_{0i}$ are on two different sides of the threshold $\alpha$.

Using the event $M_i$, the total number of misclassifications can be expressed as $M = \sum_{i=1}^{m} \mathbb{I}_{M_i}$, where $\mathbb{I}$ is the indicator function. For an allocation of $k = (k_1, \ldots, k_m)$ samples to all hypotheses, the expected total number of false decisions is given by

$$\begin{aligned} g(k) := \mathbb{E}(M|k) &= \sum_{i=1}^{m} \mathbb{P}(M_i|k_i) \\ &= \sum_{i=1}^{m} \left[\mathbb{P}(S_i/k_i > \alpha|p_i^*) \cdot \mathbb{I}(p_i^* \leq \alpha) + \mathbb{P}(S_i/k_i \leq \alpha|p_i^*) \cdot \mathbb{I}(p_i^* > \alpha)\right], \end{aligned}$$

where $S_i \sim \text{Binomial}(k_i, p_i^*)$. The aim of this chapter is to solve

$$\min_{k \in \mathbb{N}^m} g(k) \quad \text{subject to} \sum_{i=1}^{m} k_i = K. \tag{5.1}$$

The function $g$ goes to zero as $\min_{i=1,\ldots,m} k_i \to \infty$. This is to be expected as by the Law of Large Numbers, the estimates converge to the p-values as more samples are drawn, hence the probability for a wrong classification decreases.

### 5.2.2 Deriving the optimal allocation for a normal approximation

The vector $k$ minimising $g$ under the condition $\sum_{i=1}^{m} k_i = K$ is determined using a Kuhn and Tucker (1951) constraint. As derivatives are needed for the Kuhn and

Tucker (1951) formulation, the binomial distribution in $g$ is replaced by its normal approximation

$$g(k) \approx \sum_{i=1}^{m} \left[ \left( 1 - \Phi\left( \frac{k_i(\alpha - p_i^*)}{\sqrt{k_i p_i^*(1-p_i^*)}} \right) \right) \cdot \mathbb{I}(p_i^* \leq \alpha) \right.$$
$$\left. + \Phi\left( \frac{k_i(\alpha - p_i^*)}{\sqrt{k_i p_i^*(1-p_i^*)}} \right) \cdot \mathbb{I}(p_i^* > \alpha) \right] =: \gamma(k),$$

where $\Phi$ is the cumulative distribution function of the standard normal distribution. The derivative of $\gamma$ is given by

$$\frac{\partial \gamma}{\partial k_i} = \frac{p_i^* - \alpha}{2\sqrt{k_i p_i^*(1-p_i^*)}} \phi\left( \frac{k_i(\alpha - p_i^*)}{\sqrt{k_i p_i^*(1-p_i^*)}} \right) \cdot \mathbb{I}(p_i^* \leq \alpha)$$
$$+ \left( - \frac{p_i^* - \alpha}{2\sqrt{k_i p_i^*(1-p_i^*)}} \right) \phi\left( \frac{k_i(\alpha - p_i^*)}{\sqrt{k_i p_i^*(1-p_i^*)}} \right) \cdot \mathbb{I}(p_i^* > \alpha) \approx \frac{\partial g}{\partial k_i},$$

where $\phi$ is the density function of the standard normal distribution. Each partial derivative $\partial\gamma/\partial k_i$ depends on $k_i$ only.

The function $\gamma$ needs to be optimised under the constraints $k > 0$ and $\sum_{i=1}^{m} k_i = K$, meaning that each hypothesis receives at least one sample and that the total number of samples allocated equals $K$. The optimal solution $k^*$ minimising $\gamma$ satisfies

$$\nabla\gamma(k^*) = \sum_{i=1}^{m} \mu_i \nabla u_i(k^*) + \lambda \nabla v(k^*), \tag{5.2}$$

for a suitable $\lambda$, where $u_i(k) = -k_i$ and $v(k) = K - \sum_{i=1}^{m} k_i$ (Kuhn and Tucker, 1951). The functions $u_i$ encode the constraints $k_i > 0$ (primal feasibility) with $\mu_i \geq 0$ (dual feasibility) and satisfy $\mu_i u_i(k^*) = 0$ (complementary slackness), where $i \in \{1, \ldots, m\}$.

Complementary slackness and the condition $k_i > 0$ imply that $\mu_i = 0$ for all

$i \in \{1, \dots, m\}$. As each partial derivative $\partial\gamma/\partial k_i$ only depends on $k_i$, the problem simplifies to finding a $k^*$ such that $\partial\gamma/\partial k_i(k_i^*) = -\lambda^*$ for $i \in \{1, \dots, m\}$ and a $\lambda^* \geq 0$ such that $v(k^*) = 0$.

### 5.2.3 Solving for the optimal allocation

This section looks at useful computational considerations for solving (5.2). The optimal allocation for a normal approximation is obtained by tuning the parameter $\lambda$ towards an optimal value $\lambda^*$ with the property that the corresponding optimal vector $k^* = (k_1^*, \dots, k_m^*)$ solving (5.2) satisfies $v(k^*) = 0$.

The optimal $\lambda^*$ can be found using a binary search as follows. Similarly to $g$, the function $\gamma$ is positive and monotonically decreases to zero as $\min_{i=1,\dots,m} k_i \to \infty$. The derivatives $\partial\gamma/\partial k_i$ are therefore negative for $k_i > 0$ and monotonically increase to zero as $k_i \to \infty$. As a consequence, when solving $\partial\gamma/\partial k_i(k_i) = -\lambda$ for $k_i$, smaller values of $\lambda$ correspond to larger values of $k_i$ and vice versa.

This observation applies to all $k_i$, $i \in \{1, \dots, m\}$. Thus $\sum_{i=1}^{m} k_i$ also monotonically decreases for each vector $k$ solving (5.2) as $\lambda \to \infty$. The optimal value can hence be found easily with a binary search. Likewise, as $\gamma$ is monotonic, the solution $k_i$ of $\partial\gamma/\partial k_i(k_i) = -\lambda$ can be found with a binary search in $k_i$ for each proposed value of $\lambda$.

## 5.3 Empirical comparison of `QuickMMCTest` to the optimal allocation of samples

### 5.3.1 The `QuickMMCTest` algorithm

A runtime study conducted in this section indicates that the allocation of samples to multiple hypotheses computed by the `QuickMMCTest` algorithm (Chapter 4) might be asympotically close to the optimal allocation derived under the normal approximation.

`QuickMMCTest` is used to iteratively allocate samples to all $m$ hypotheses. The algorithm is designed in such a way as to allocate more samples adaptively to hypotheses with p-values which are closer to the testing threshold and thus harder to classify. Although `QuickMMCTest` is capable, in principle, of computing allocations for tests evaluated with arbitrary step-up and step-down procedures, we use it in connection with the Bonferroni (1936) correction $h(p, \alpha)$ only (Section 5.2.1). Moreover, `QuickMMCTest` depends on two parameters: the total number of samples $K$ to be spent and the total number of iterations $n_{\max}$. The choice of $K$ and $n_{\max}$ is given individually in each of the following subsections.

### 5.3.2 Empirical comparison to the optimal allocation

The allocation of samples computed by `QuickMMCTest` will be compared to the real-valued optimal allocation derived in Section 5.2. Although we use the real-valued optimal allocation as a reference for the comparison, a greedy integer solution obtained in Section 5.4 indicates that the integer and real-valued optimal solutions might not considerably differ.

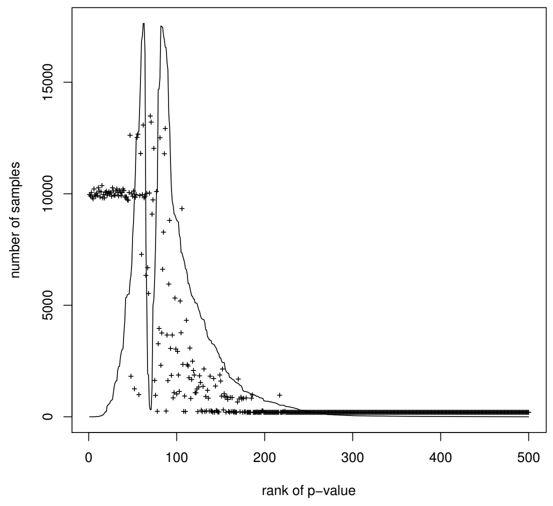


Figure 5.1: Real-valued optimal solution (solid line) computed via Kuhn-Tucker constraint and allocation of samples returned by `QuickMMCTest` (crosses).

For all comparisons in this section, we use the mixture distribution of Sandve et al. (2011) introduced in Section 4.3. We fix a realisation generated from this mixture distribution with $m = 500$ p-values and parameter $\pi_0 = 0.5$. The p-values were then sorted in order to be able to plot the real-valued optimal allocation as a smooth curve and hence to increase clarity in plots. The fixed p-values will be referred to as $p_i^*$, $i \in \{1, \dots, m\}$.

We compute the optimal real-valued allocation for our fixed distribution as described in Section 5.2.3. Additionally, we apply `QuickMMCTest` to the fixed p-values and record its allocation of samples to each hypothesis. Testing was carried out using a Bonferroni (1936) threshold of $\alpha = 0.1/m$.

Figure 5.1 shows the real-valued optimal solution (solid curve) as well as the number of samples allocated to each hypothesis by `QuickMMCTest` (crosses). The total number of samples was $K = 10^6$. The cutoff for multiple testing (the last p-value in the rejection region of the test) occurred at the 80th p-value.

As visible from the real-valued optimal allocation in Figure 5.1, as expected, p-values far away from the cutoff point and thus far away from the threshold are not allocated many samples. On approaching the cutoff, more samples are needed to accurately classify which p-values belong to the rejection and which belong to the non-rejection region of the testing procedure. However, p-values which are too close to the threshold on either side are not worth being allocated too many samples as deciding them is out of scope for a limited effort (see Section 2.6.1 and Section 2.6.2). Therefore, the real-valued optimal allocation decreases on approaching the closest p-value to the threshold from both sides, yielding a bimodal allocation.

The allocation computed by `QuickMMCTest` roughly resembles the shape of the optimal allocation in the sense that the optimal allocation seems to be some kind of rough hull curve for the allocation of `QuickMMCTest`. Both the optimal allocation and the one of `QuickMMCTest` fairly coincide for p-values considerably above the threshold. Moreover, `QuickMMCTest` allocates samples to hypotheses with p-values above the threshold in such a way that its allocation roughly approximates the second mode of the real-valued allocation. However, the first mode of the real-valued allocation is poorly approximated as `QuickMMCTest` seems to give an equal number of samples (according to Figure 5.1: about 10000 samples) to each hypothesis below the cutoff.

### 5.3.3 Asympotic behaviour of the allocation of samples

We investigate how the real-valued optimal allocation and the one of `QuickMMCTest` behave as the total number $K$ of samples to be spent increases.

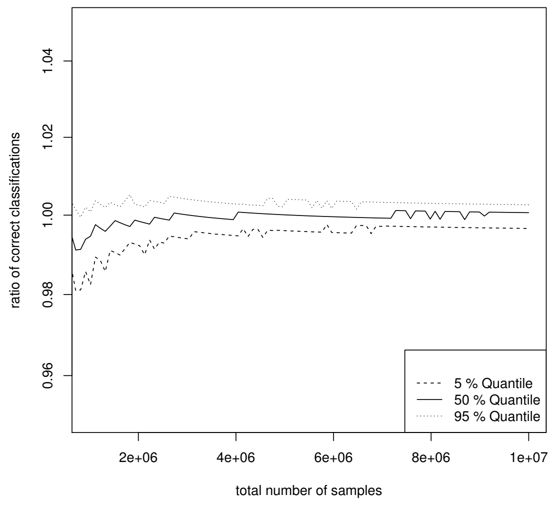


Figure 5.2: Indication of convergence of `QuickMMCTest`'s solution to the optimal solution. 5%, 50% and 95%-quantiles of the number of correctly classified hypotheses of `QuickMMCTest` divided by the expected number of correctly classified hypotheses.

This is done using the fixed set of $m = 500$ p-values for a varying total number of samples $K$ which was increased from $10^5$ to $10^7$ in 100 steps. We calculate 5%, 50% and 95% quantiles of the empirical number of correctly classified hypotheses for `QuickMMCTest` divided by the expected number of correctly classified hypotheses (based on $r = 10$ repetitions only as computing the optimal allocation is very costly for large total sample numbers). The expected number of correctly classified hypotheses is obtained by evaluating $\gamma$ on the optimal allocation derived in Section 5.2.2 and by subtracting the resulting value from $m$. A plot is given in Figure 5.2.

Figure 5.2 indicates that the ratio of correct classifications of `QuickMMCTest` to the ones of the optimal allocation might converge to one as $K \to \infty$. Based on Figure 5.1 and 5.2, one could conjecture that the allocation returned by `QuickMMCTest` behaves in such a way as to mimic the optimal one in terms of correctly classified hypotheses.

## 5.4 A greedy solution to the integer version of the optimal allocation problem

We investigate one of many possible greedy algorithms which tries to compute a discrete optimal allocation for a given set of p-values, thus attempting to solve (5.1) directly.

The aim of this section is to argue that, although unknown, the optimal integer allocation is most likely of a similar form as the real-valued optimal solution derived in Section 5.2.

### 5.4.1 A greedy algorithm

For notational convenience, the function $g$ introduced in Section 5.2 is expressed as $g(k) = \sum_{i=1}^{m} g_i(k_i)$ for $k = (k_1, \dots, k_m)$ using functions

$$g_i(k_i) = \mathbb{P}(M_i|k_i) = \mathbb{P}(S_i/k_i > \alpha|p_i^*) \cdot \mathbb{I}(p_i^* \leq \alpha) + \mathbb{P}(S_i/k_i \leq \alpha|p_i^*) \cdot \mathbb{I}(p_i^* > \alpha),$$

where $M_i = \{S_i/k_i > \alpha \wedge p_i^* \leq \alpha\} \cup \{S_i/k_i \leq \alpha \wedge p_i^* > \alpha\}$ and $S_i \sim \text{Binomial}(k_i, p_i^*)$ (see Section 5.2). We consider the following greedy approach (Algorithm 5.1) to compute a discrete optimal allocation for a given vector of p-values $p^* = (p_1^*, \dots, p_m^*)$.

**Algorithm 5.1:** Greedy algorithm to compute an integer allocation of samples

```
input: $p$, $\alpha$, $K$
1 $jump \leftarrow 1/\alpha$; $j \leftarrow 1$; $b_i \leftarrow 0$, $k_i \leftarrow \mathbb{I}(p_i^* \leq \alpha)$, $i \in \{1, \dots, m\}$;
2 while $\sum_{i=1}^m k_i + b_j < K$ do
3     $k_j \leftarrow k_j + b_j$;
4     for $i \leftarrow 1$ to $m$ do
5         $b_i \leftarrow \begin{cases} \min\{z \in \mathbb{N} : g_i(k_i + z) < g_i(k_i)\} & p_i > \alpha, \\ jump - 1 & p_i \leq \alpha, k_i < jump, \\ jump & p_i \leq \alpha, k_i \geq jump. \end{cases}$
6         $d_i \leftarrow g_i(k_i + b_i) - g_i(k_i)$;
7     $j \leftarrow \arg\max_{i=1,\dots,m} d_i/b_i$;
8 return $(k_1, \dots, k_m)$;
```

Apart from the p-values, the greedy algorithm also requires the testing threshold $\alpha$ and the total number $K$ of samples to be allocated to the $m$ hypotheses under consideration.

In order to highlight the reasoning behind Algorithm 5.1, consider the behaviour of the function $g_i$ for a p-value $p_i^*$ below and above the threshold $\alpha$ as illustrated exemplarily in Figure 5.3. We used $m = 500$ p-values and a threshold $\alpha = 0.1/m = 1/5000$. Figure 5.3 displays the expected number of misclassifications, obtained by evaluating $g_i$, against the number of samples for a hypothesis below (left) and above (right) the threshold. Drawing no samples is defined to yield a p-value of zero and hence a correct rejection if the hypothesis corresponding to that p-value is rejected.

As shown in Figure 5.3, the behaviour of $g_i$ exhibits a particular structure caused by the following effect. A rejection is obtained if a p-value estimate is below the

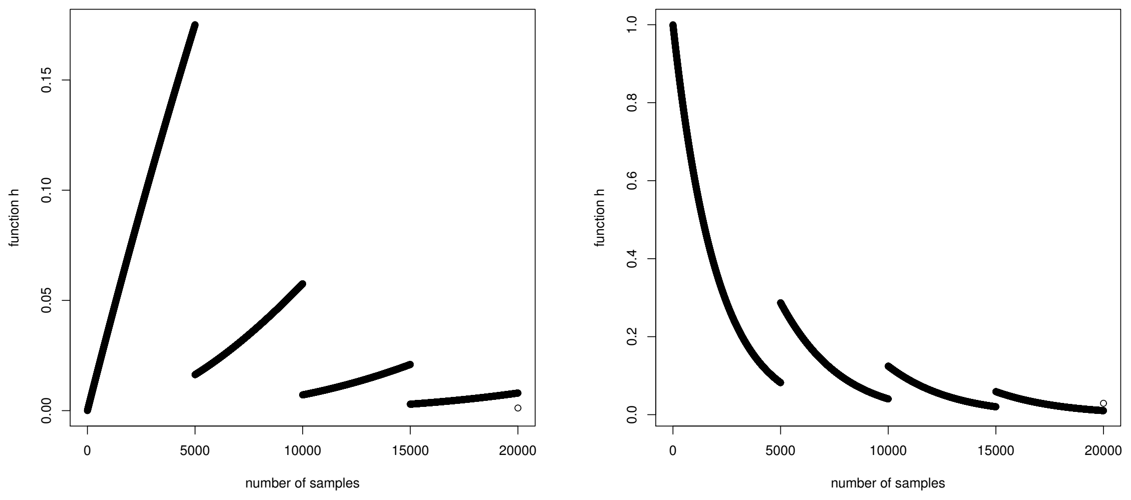


Figure 5.3: Example showing the expected number of misclassifications (obtained by evaluating the function $g_i$) against number of samples for a hypothesis below (left) and above (right) the threshold.

threshold at $\alpha = 1/5000$. Thus for a p-value below $\alpha$ (left plot), when using less than 5000 samples, a p-value will be classified as rejected only if 0 exceedances are observed. Consequently, using that no samples yield a p-value of zero, drawing more samples only potentially increases the probability of making a false decision. When reaching $1/\alpha = 5000$ samples, observing both 0 exceedances or 1 exceedance will lead to a rejection and hence to a correct decision. The expected number of misclassifications drops. This effect repeats itself every $1/\alpha$ samples.

For a p-value above $\alpha$ (right plot), the inverse effect happens. Drawing no samples leads to a rejection (a p-value of 0) and thus to a sure misclassification. Drawing more samples decreases the expected number of misclassifications as observing 0 exceedances out of $k_i$ samples (the only case in which the hypothesis will be rejected and thus misclassified) becomes less and less likely as $k_i$ approaches $1/\alpha = 5000$.

When reaching $1/\alpha = 5000$ samples, as before, observing both 0 exceedances and 1 exceedance lead to a rejection and thus to a misclassification. The expected number of misclassifications increases again.

As expected, the probability for a misclassification of an hypothesis $H_{0i}$ vanishes in both cases as $k_i \to \infty$.

Algorithm 5.1 works by greedily trying to fill up a zero vector $k$ storing the final allocation of samples in such a way as to decrease the function $g$ in every step, subject to the constraint $\sum_{i=1}^{m} k_i = K$. In each iteration, Algorithm 5.1 individually determines a sensible number of $b_i$ samples to be allocated to each hypothesis $H_{0i}$, $i \in \{1, \ldots, m\}$. This number varies for each hypothesis: for hypotheses above the threshold, $b_i$ is determined as the smallest increase that leads to a decrease in $g$. For hypotheses below the threshold, only "jumping" by an amount of $(1/\alpha - 1)$ samples is sensible if $k_i = 1$ or jumping $1/\alpha$ samples in the case $k_i = 0$: that is, Algorithm 5.1 jumps to the left end of each of the branches in the left plot of Figure 5.3.

Certain jumps might lead to a large decrease in the value of the function $g$, but only at the cost of also spending (too) many samples on just one hypothesis. Algorithm 5.1 therefore determines the decrease $d_i$ in function $g$ for each $H_{0i}$ associated to spending $b_i$ samples and allocates the batch of $b_i$ samples to the hypothesis yielding the best ratio $d_i/b_i$.

Algorithm 5.1 is just one of many possible greedy approaches to compute a sensible integer allocation of samples. We tried various variants and allocation rules, all with a similar behaviour as Algorithm 5.1 and resulting in similar allocations.

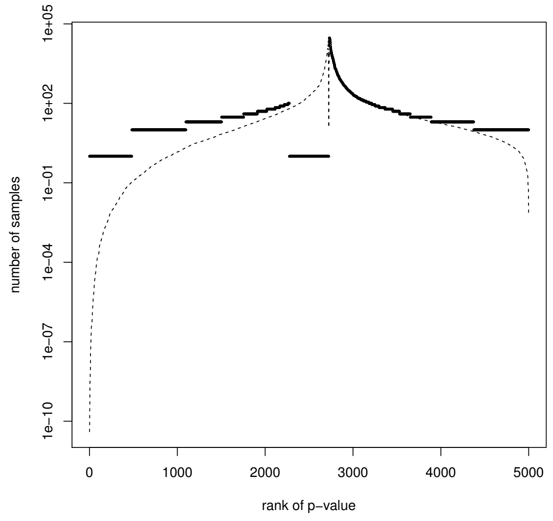


Figure 5.4: Greedy solution of Algorithm 5.1 (bold) compared to the real-valued optimal solution (dotted) computed as described in Section 5.2.3. Log scale on the y-axis.

### 5.4.2 Comparison of the greedy integer solution to the optimal real-valued solution

Figure 5.4 compares two allocations, the greedy discrete one computed by Algorithm 5.1 (bold) and the real-valued optimal one computed via Kuhn and Tucker (1951) constraint (dotted) as described in Section 5.2.3.

For Figure 5.4, a new realisation of $m = 5000$ p-values was drawn from the mixture distribution introduced in Section 5.3.2 with parameter $\pi_0 = 0.5$. Testing was carried out at a constant (uncorrected, that is not divided by $m$) threshold of 0.1.

Figure 5.4 shows that the two allocations are qualitatively similar, in particular they largely agree for hypotheses having p-values above the threshold (the threshold intersects the p-value distribution around rank 2900). The allocations are also qualitatively similar for p-values below the threshold even though Algorithm 5.1 allocates slightly more samples than the optimal allocation in this case. This discrepancy increases as the p-values tend to zero (to be precise, to low ranks in Figure 5.4). However, as seen in the plot, the optimal allocation distributes fractions of a sample to low p-values. These would, for instance, be rounded up to one sample per hypothesis when drawing samples in practice, thus making the greedy discrete and the optimal real-valued allocations coincide.

The optimal real-valued allocation and the greedy one of Algorithm 5.1 differ around the threshold. Strikingly, the greedy solution does not allocate more samples than the pre-set one initial sample to rejected hypotheses close to the threshold (to be precise, it allocates one sample each over a rather large interval to the left of the threshold), whereas the real-valued Kuhn and Tucker (1951) solution allocates samples roughly symmetrically and very confined around the threshold.

## 5.5 Discussion

Many algorithms published in the literature are designed to evaluate multiple hypotheses using Monte Carlo simulations. Though sensible, it is unclear if these algorithms are able to draw samples in an optimal way defined, for instance, as minimising the expected number of misclassified hypotheses.

The present chapter is concerned with the allocation of samples which minimises the expected number of misclassified hypotheses. We derive the optimal allocation of a finite number of samples under the assumption that the p-values are known and that the number of samples is real-valued using a Kuhn and Tucker (1951) constraint and a normal approximation. Though all simulations and conclusions are based on the optimal solution assuming real and not integer values for the number of samples, we demonstrate exemplarily that the real-valued solution derived under a normal approximation is qualitatively similar to a greedy integer solution, indicating that the real-valued and the unknown optimal integer solution might be similar.

A simulation study indicates that the allocation of samples to each hypothesis computed by the simple `QuickMMCTest` algorithm (Chapter 4) might behave roughly similarly to the optimal allocation of samples. In contrast to the optimal allocation which was derived using full knowledge of all p-values, `QuickMMCTest` does not require knowledge of any p-value as long as it is possible to obtain samples under each null hypothesis.

# 6 Conclusion

The present thesis investigates multiple testing from several perspectives. The theme connecting all chapters is the assumption that several hypotheses are to be tested for statistical significance without knowledge of the p-values underlying the tests. Instead, all p-values have to be approximated via Monte Carlo simulation, precisely by drawing samples for each hypothesis under the null. Such a scenario is commonplace is most applications involving real biological datasets.

The first part of this thesis focuses on the computation of test results with a guarantee on their correctness in the aforementioned scenario, that is test results which are identical to the ones obtained if all p-values had been available analytically. Chapter 2 offers a solution to this problem by presenting `MMCTest`, an algorithm to implement a multiple testing procedure. In contrast to algorithms published in the literature, Chapter 2 proves that up to a pre-specified error probability, `MMCTest` yields correct decisions on all hypotheses based solely on Monte Carlo simulation. By exemplarily considering the classification of a real gene expression dataset, we demonstrate that `MMCTest` can be used to reveal the (previously unknown) correct decision (significant, non-significant) of certain genes of interest (up to the error probability), a unique feature not provided by other methods.

The ideas behind `MMCTest` can be generalised and applied to a whole range of existing algorithms available in the literature. Chapter 3 demonstrates that existing methods giving no guarantee on their test result can be modified to yield theoretical guarantees on the correctness of their outputs. Chapter 2 and 3 together thus provide a means of achieving Monte Carlo based multiple hypothesis testing with guarantees on the correctness of all test results.

In practice, it might be desired to sacrifice the computational effort needed to obtain a guaranteed test result and to invest it instead in the computation of an ad-hoc test result with a high accuracy. Here, the accuracy of a test result is measured in terms of misclassifications, that is decisions on single hypotheses which are different from the ones obtained if the p-values underlying the tests had been available. Chapter 4 presents an algorithm for this scenario based on Thompson (1933) Sampling, called `QuickMMCTest`. The algorithm achieves an excellent accuracy in a simulation study on a variety of multiple testing procedures and it offers a solution to Monte Carlo based multiple testing without the use of discrete p-value estimates, a desirable feature for practical use as it circumvents a phenomenon observed in published methods leading to meaningless results consisting of non-rejected hypotheses only.

Chapter 5 demonstrates empirically that `QuickMMCTest` might behave close to being optimal in the sense that its allocation roughly mimics the optimal allocation of samples, understood as the one minimising the expected number of misclassifications. For this, the optimal allocation of a finite number of samples is derived under a normal approximation and the assumption that multiple testing is carried out using the Bonferroni (1936) correction.

# 7 Future work

The topics dealt with in this thesis leave considerable scope for further improvements and extensions.

Several topics related to the `MMCTest` algorithm (Chapter 2) are still unsolved or leave scope for extensions. First, a detailed analysis of the runtime of `MMCTest` is still pending. Although it is proven that the expected runtime for a complete classification is infinite, it would be of great interest to explore further under which circumstances and for which procedures a finite runtime can be achieved. This could be attempted by analysing the speed with which the length of certain confidence intervals goes to zero. Deriving a precise runtime analysis will be challenging as the runtime of `MMCTest` depends both on the multiple testing procedure used as well as on the (potentially unknown) p-value distribution.

Chapter 3 extends the idea behind `MMCTest` by presenting a framework for multiple testing. This framework can be extended to ever more multiple testing situations, for instance to the emerging topic of testing in directed acyclic graphs (Goeman and Mansmann, 2008). Such scenarios naturally occur in hierarchical multiple testing problems having a natural ordering of the tests, for instance due to subset relations (SNPs, genes, chromosomes). Usually, a node hypothesis is true if all its children are true, and the truth of any node implies the truth of all its children. Hypotheses in an acyclic directed graph can be tested, for instance, using a bottom-up or top-down approach, and the overall testing threshold can be distributed in a variety of ways. Also, if a combined hypothesis is false, then at least one child hypothesis must be false which opens up the possibility for various shortcuts to avoid actually carrying out all tests.

Moreover, new approaches to multiple testing have emerged recently, such as the one of Goeman and Solari (2011): the authors reverse the classical roles in multiple testing by proposing to choose the set of rejections freely and provide a method based on modified closed testing procedures which gives a confidence statement on the number of false rejections. It will be interesting to extend the ideas behind the `MMCTest` algorithm to such scenarios and provide guarantees on the number of erroneous decisions when p-values have to be approximated in a Monte Carlo scenario.

Second, this thesis also views multiple testing from a practical point of view: Chapter 4 proposes an algorithm called `QuickMMCTest` based on Thompson (1933) Sampling. Empirical results demonstrate that such an approach to fine-tune the allocation of samples results in a superior accuracy compared to published methods. Although the method has been tested on several thousand hypotheses, it remains to investigate to which extent it is scalable to large scale multiple testing problems, to apply it to large scale multiple testing and to enhance the method if needed.

Importantly, Chapter 5 demonstrates empirically that `QuickMMCTest` seems to mimic the optimal allocation of samples when classifying hypotheses using the Bonferroni (1936) correction (assuming that the analytical p-values are known and hence the optimal allocation of samples can be calculated). Here, the optimal (real-valued) allocation is understood as the one minimising the expected number of erroneously classified hypotheses when numbers of samples are allowed to take real as opposed to integer values, derived via Kuhn and Tucker (1951) constraint. A rigorous proof of this optimality statement is still pending and would greatly underpin the `QuickMMCTest` approach. I would like to attempt this by analysing by how much the current allocation of `QuickMMCTest` in each iteration differs from the optimal Kuhn and Tucker (1951) solution.

Assuming full knowledge of all p-values in order to compute the optimal allocation of samples, it will be straightforward to demonstrate that the `QuickMMCTest` algorithm also asymptotically mimics the optimal allocation when using any arbitrary step-up or step-down procedure to correct for multiple tests. This is due to the fact that the testing result of any step-up or step-down procedure can also be obtained using the Bonferroni (1936) correction applied with the p-value of the last rejected hypothesis as constant and known threshold. As a derivation of the discrete (integer-valued) optimal allocation seems infeasible (due to the fact that the problem is non-convex), one approach to demonstrate optimal behaviour of `QuickMMCTest` could be to prove that any optimum derived via Kuhn and Tucker (1951) formalism is also (one of the) optima of the discrete allocation: this could, in principle, be shown using subdifferential versions of the Kuhn and Tucker (1951) conditions (Ruszczynski, 2006).

Multiple testing is traditionally carried out by merely differentiating between significant and non-significant hypotheses. However, the traditional approach might not always be desired when evaluating real data studies. Instead, ranking hypotheses based on their p-values or merely obtaining the $k$ lowest p-values has immediate advantages as it allows to distinguish between more and less *interesting* hypotheses. The parameter $k$ is often given as a limiting factor in practice: $k$ could be, for instance, the maximal number of genes which can be included in follow-up studies due to budget constraints. In this case, one wishes to select the most promising hypotheses, meaning the ones having the lowest p-values, independently of their actual testing result (significant or non-significant). In fact, recent research has shown that hypotheses with a low p-value were worth being included in follow-up studies even though they were not significant. For instance, Xu and Taylor (2009) report that a SNP considered in a prostate cancer study had an initial p-value (Yeager et al., 2007) of "only 0.042, but the P-value was $7.31 \cdot 10^{-13}$ in a follow up study" of Thomas et al. (2008). I would like to investigate this approach further following the spirit of `MMCTest` by providing a method that gives, with pre-specified probability chosen by the user, a set that is equal to or that is guaranteed to contain the $k$ most significant p-values. This can be achieved by excluding hypotheses successively from the target set using confidence intervals on all p-values.

# References

Agrawal, S. and Goyal, N. (2012). Analysis of Thompson Sampling for the Multi-armed Bandit Problem. *JMLR: Workshop and Conference Proceedings of the 25th Annual Conference on Learning Theory*, 23(39):1–26.

Asomaning, N. and Archer, K. (2012). High-throughput dna methylation datasets for evaluating false discovery rate methodologies. *Comput Stat Data An*, 56:1748–1756.

Benjamini, Y. and Hochberg, Y. (1995). Controlling the false discovery rate: A practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society, Series B (Methodological)*, 57(1):289–300.

Benjamini, Y. and Hochberg, Y. (2000). On the adaptive control of the false discovery rate in multiple testing with independent statistics. *J. Edu. Behav. Stats.*, 25(1):60–83.

Benjamini, Y. and Yekutieli, D. (2001). The control of the false discovery rate in multiple testing under dependency. *The Annals of Statistics*, 29(4):1165–1188.

Besag, J. and Clifford, P. (1991). Sequential Monte Carlo p-values. *Biometrika*, 78(2):301–304.

Bonferroni, C. (1936). Teoria statistica delle classi e calcolo delle probabilità. *Pubblicazioni del R Istituto Superiore di Scienze Economiche e Commerciali di Firenze*, 8:3–62.

Chen, J., Bushman, F., Lewis, J., Wu, G., and Li, H. (2013). Structure-constrained sparse canonical correlation analysis with an application to microbiome data analysis. *Biostatistics*, 14(2):244–258.

Chen, W. and Dieckmann, C. (1997). Genetic evidence for interaction between Cbp1 and specific nucleotides in the 5' untranslated region of mitochondrial cytochrome b mRNA in Saccharomyces cerevisiae. *Mol Cell Biol*, 17(11):6203–6211.

Cheng, C. (2006). An adaptive significance threshold criterion for massive multiple hypotheses testing. *IMS Lecture Notes – Monograph Series of the 2nd Lehmann Symposium – Optimality*, 49:51–76.

Cheng, C. (2009). Internal validation inferences of significant genomic features in genome-wide screening. *Computational Statistics and Data Analysis*, 53:788–800.

Cherry, J., Hong, E., Amundsen, C., Balakrishnan, R., Binkley, G., Chan, E., Christie, K., Costanzo, M., Dwight, S., Engel, S., Fisk, D., Hirschman, J., Hitz, B., Karra, K., Krieger, C., Miyasato, S., Nash, R., Park, J., Skrzypek, M., Simison, M., Weng, S., and Wong, E. (2011). Saccharomyces Genome Database: the genomics resource of budding yeast. *Nucleic Acids Res*, 40(D):700–705.

Clopper, C. and Pearson, E. (1934). The use of confidence or fiducial limits illustrated in the case of the binomial. *Biometrika*, 26(4):404–413.

Cohen, O., Ashkenazy, H., Burstein, D., and Pupko, T. (2012). Uncovering the co-evolutionary network among prokaryotic genes. *Bioinformatics*, 28(ECCB):i389–i394.

David, N. and Nagaraja, H. (2003). *Order Statistics*. Wiley.

Davison, A. and Hinkley, D. (1997). *Bootstrap Methods and Their Application*. Cambridge University Press.

Dazard, J.-E. and Rao, S. (2012). Joint adaptive meanvariance regularization and variance stabilization of high dimensional data. *Comput Stat Data An*, 56:2317–2333.

DDB (2001). Gene Ontology annotation through association of InterPro records with GO terms. *unpublished*.

Edgington, E. and Onghena, P. (1997). *Randomization tests*. Fourth Edition, Chapman & Hall/CRC, Boca Raton, FL.

Finner, H. and Gontscharuk, V. (2009). Controlling the familywise error rate with plug-in estimator for the proportion of true null hypotheses. *Journal of the Royal Statistical Society: Series B (Methodological)*, 71(5):1031–1048.

Fox, T. (1979). Genetic and physical analysis of the mitochondrial gene for subunit II of yeast cytochrome c oxidase. *J Mol Biol*, 130(1):63–82.

Gandy, A. (2009). Sequential implementation of Monte Carlo tests with uniformly bounded resampling risk. *Journal of the American Statistical Association*, 104(488):1504–1511.

Gandy, A. and H., G. (2014). MMCTest – A Safe Algorithm for Implementing Multiple Monte Carlo Tests. *Scandinavian Journal of Statistics*, 41(4):1083–1101.

Gandy, A. and H., G. (2015a). A framework for Monte-Carlo based multiple testing. *arXiv:1402.3019*.

Gandy, A. and H., G. (2015b). Optimal allocation of samples for Monte-Carlo based multiple testing and comparison to Thompson Sampling. *arXiv:1502.07864*.

Gandy, A. and H., G. (2015c). QuickMMCTest – Higher accuracy for Monte-Carlo based multiple testing. *arXiv:1402.2706*.

Gleser, L. (1996). Comment on 'Bootstrap Confidence Intervals' by T. J. DiCiccio and B. Efron. *Statist. Sci.*, 11:219–221.

Goeman, J. and Mansmann, U. (2008). Multiple testing on the directed acyclic graph of gene ontology. *Bioinformatics*, 24(4):537–544.

Goeman, J. and Solari, A. (2011). Multiple testing for exploratory research. *Statistical Science*, 26(4):584–597.

Guo, W. and Peddada, S. (2008). Adaptive choice of the number of bootstrap samples in large scale multiple testing. *Stat Appl Genet Mol Biol.*, 7(1):1–16.

Gusenleitner, D., Howe, E., Bentink, S., Quackenbush, J., and Culhane, A. (2012). iBBiG: iterative binary bi-clustering of gene sets. *Bioinformatics*, 28(19):2484–2492.

Han, B. and Dalal, S. (2012). A Bernstein-type estimator for decreasing density with application to p-value adjustments. *Computational Statistics and Data Analysis*, 56:427–437.

Hochberg, Y. (1988). A sharper Bonferroni procedure for multiple tests of significance. *Biometrika*, 75(4):800–802.

Hoeffding, W. (1963). Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30.

Holm, S. (1979). A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6(2):65–70.

Hommel, G. (1988). A stagewise rejective multiple test procedure based on a modified Bonferroni test. *Biometrika*, 75(2):383–386.

Hwang, Y.-T. (2011). Comparisons of estimators of the number of true null hypotheses and adaptive fdr procedures in multiplicity testing. *Computational Statistics and Data Analysis*, 81(2):207–220.

Jiang, H. and Doerge, R. (2008). Estimating the proportion of true null hypotheses for multiple comparisons. *Cancer Informatics*, 6:25–32.

Jiang, H. and Salzman, J. (2012). Statistical properties of an early stopping rule for resampling-based multiple testing. *Biometrika*, 99(4):973–980.

Johnson, V. (2013). Revised standards for statistical evidence. *Proc Natl Acad Sci*, 110(48):19313–19317.

Jupiter, D., Sahutoglu, J., and VanBuren, V. (2010). TreeHugger: A new test for enrichment of gene ontology terms. *INFORMS Journal on Computing*, 22(2):210–221.

Kim, S.-B., Yang, S., Kim, S.-K., Kim, S., Woo, H., Volsky, D., Kim, S.-Y., and Chu, I.-S. (2007). GAzer: gene set analyzer. *Bioinformatics*, 23(13):1697–1699.

Knijnenburg, T., Wessels, L., Reinders, M., and Shmulevich, I. (2009). Fewer permutations, more accurate p-values. *Scandinavian Journal of Statistics*, 25(12):i161–i168.

Kreike, J., Bechmann, H., Van Hemert, F., Schweyen, R., Boer, P., Kaudewitz, F., and Groot, G. (1979). The Identification of Apocytochrome b as a Mitochondrial Gene Product and Immunological Evidence for Altered Apocytochrome b in Yeast Strains having Mutations in the COB Region of Mitochondrial DNA. *Eur J Biochem*, 101(2):607–617.

Kuhn, H. and Tucker, A. (1951). Nonlinear Programming. *Proc. Second Berkeley Symp. on Math. Statist. and Prob.*, pages 481–492.

Lai, T. (1976). On Confidence Sequences. *Ann. Statist.*, 4(2):265–280.

Langaas, M., Lindqvist, B., and Ferkingstad, E. (2005). Estimating the proportion of true null hypotheses, with application to dna microarray data. *Journal of the Royal Statistical Society: Series B (Methodological)*, 67(4):555–572.

Li, G., Best, N., Hansell, A., Ahmed, I., and Richardson, S. (2012). BaySTDetect: detecting unusual temporal patterns in small area data via bayesian model choice. *Biostatistics*, 13(4):695–710.

Li, J., Tai, B., and Nott, D. (2009). Confidence interval for the bootstrap p-value and sample size calculation of the bootstrap test. *Journal of Nonparametric Statistics*, 21(5):649–661.

Lin, D. (2005). An efficient Monte Carlo approach to assessing statistical significance in genomic studies. *Bioinformatics*, 21(6):781–787.

Liu, J. and Chen, R. (1998). Sequential monte carlo methods for dynamic systems. *Journal of the American Statistical Association*, 93(443):1032–1044.

Liu, J., Huang, J., Ma, S., and Wang, K. (2013). Incorporating group correlations in genome-wide association studies using smoothed group Lasso. *Biostatistics*, 14(2):205–219.

Lourenco, V. and Pires, A. (2014). M-regression, false discovery rates and outlier detection with application to genetic association studies. *Comput Stat Data An*, 78:33–42.

Lu, H.-F., Dong, H.-T., Sun, C.-B., Qing, D.-J., Li, N., Wu, Z.-K., Wang, Z.-Q., and Li, Y.-Z. (2011). The panorama of physiological responses and gene expression of whole plant of maize inbred line yq7-96 at the three-leaf stage under water deficit and re-watering. *Theor Appl Genet*, 123:943–958.

Manly, B. (1997). *Randomization, bootstrap and MonteCarlo methodsin biology.* Second Edition, Chapman & Hall, London.

Martínez-Camblor, P. (2014). On correlated z-values distribution in hypothesis testing. *Comput Stat Data An*, 79:30–43.

Meinshausen, N. (2006). False discovery control for multiple tests of association under general dependence. *Scandinavian Journal of Statistics*, 33(2):227–237.

Meunier, B. (2001). Site-directed mutations in the mitochondrially encoded subunits I and III of yeast cytochrome oxidase. *Biochem J*, 354(2):407–412.

Nusinow, D., Kiezun, A., O'Connell, D., Chick, J., Yue, Y., Maas, R., Gygi, S., and Sunyaev, S. (2012). Network-based inference from complex proteomic mixtures using SNIPE. *Bioinformatics*, 28(23):3115–3122.

Pekowska, A., Benoukraf, T., Ferrier, P., and Spicuglia, S. (2010). A unique h3k4me2 profile marks tissue-specific gene regulation. *Genome Research*, 20(11):1493–1502.

Phipson, B. and Smyth, G. (2010). Permutation P-values Should Never Be Zero: Calculating Exact P-values When Permutations Are Randomly Drawn. *Stat. Appl. Genet. Mol. Biol.*, 9(1):1–12.

Pounds, S. and Cheng, C. (2006). Robust estimation of the false discovery rate. *Bioinformatics*, 22(16):1979–1987.

Rahmatallah, Y., Emmert-Streib, F., and Glazko, G. (2012). Gene set analysis for self-contained tests: complex null and specific alternative hypotheses. *Bioinformatics*, 28(23):3073–3080.

Rom, D. (1990). A sequentially rejective test procedure based on a modified Bonferroni inequality. *Biometrika*, 77(3):663–665.

Romano, J. and Shaikh, A. (2006). Stepup procedures for control of generalizations of the familywise error rate. *The Annals of Statistics*, 34(4):1850–1873.

Roth, A. (1999). Multiple comparison procedures for discrete test statistics. *J Statist Plann Inference*, 82(1-2):101–117.

Rouillard, J., Dufour, M., Theunissen, B., Mandart, E., Dujardin, G., and Lacroute, F. (1996). SLS1, a new Saccharomyces cerevisiae gene involved in mitochondrial metabolism, isolated as a syntheticlethal in association with an SSM4 deletion. *Mol Gen Genet*, 252(6):700–708.

Ruszczynski, A. (2006). *Nonlinear Optimization*. Princeton University Press.

Sandve, G., Ferkingstad, E., and Nygård, S. (2011). Sequential Monte Carlo multiple testing. *Bioinformatics*, 27(23):3235–3241.

Schweder, T. and Spjøtvoll, E. (1982). Plots of p-values to evaluate many tests simultaneously. *Biometrika*, 69(3):493–502.

Shaffer, J. (1986). Modified sequentially rejective multiple test procedures. *Journal of the American Statistical Association*, 81(395):826–831.

Sidak, Z. (1967). Rectangular confidence regions for the means of multivariate normal distributions. *Journal of the American Statistical Association*, 62(318):626–633.

Simes, R. (1986). An improved Bonferroni procedure for multiple tests of significance. *Biometrika*, 73(3):751–754.

Storey, J. (2002). A direct approach to false discovery rates. *Journal of the Royal Statistical Society: Series B (Methodological)*, 64(3):479–498.

Tamhane, A. and Liu, L. (2008). On weighted Hochberg procedures. *Biometrika*, 95(2):279–294.

Thomas, G., Jacobs, K., Yeager, M., Kraft, P., Wacholder, S., Orr, N., Yu, K., Chatterjee, N., Welch, R., Hutchinson, A., Crenshaw, A., Cancel-Tassin, G., Staats, B., Wang, Z., Gonzalez-Bosquet, J., Fang, J., Deng, X., Berndt, S., Calle, E., Feigelson, H., Thun, M., Rodriguez, C., Albanes, D., Virtamo, J., Weinstein, S., Schumacher, F., Giovannucci, E., Willett, W., Cussenot, O., Valeri, A., Andriole, G., Crawford, E., Tucker, M., Gerhard, D., Fraumeni Jr, J., Hoover, R., Hayes, R., Hunter, D., and Chanock, S. (2008). Multiple loci identified in a genome-wide association study of prostate cancer. *Nature Genetics*, 40(3):310–315.

Thompson, W. (1933). On the Likelihood that One Unknown Probability Exceeds Another in View of the Evidence of Two Samples. *Biometrika*, 25(3/4):285–294.

Thulin, M. (2014). A high-dimensional two-sample test for the mean using random subspaces. *Comput. Stat. Data An.*, 74:26–38.

Tusher, V., Tibshirani, R., and Chu, G. (2001). Significance analysis of microarrays applied to the ionizing radiation response. *Proc. Natl. Acad. Sci. USA*, 98(18):5116–5121.

van Wieringen, W., van de Wiel, M., and van der Vaart, A. (2008). A test for partial differential expression. *Journal of the American Statistical Association*, 103(483):1039–1049.

Wald, A. (1945). Sequential tests of statistical hypotheses. *The Annals of Mathematical Statistics*, 16(2):117–186.

Westfall, P. and Troendle, J. (2008). Multiple testing with minimal assumptions. *Biom J.*, 50(5):745–755.

Westfall, P. and Young, S. (1993). *Resampling-based multiple testing: Examples and methods for p-value adjustment*. Wiley, New York.

Wu, H., Wang, C., and Wu, Z. (2013). A new shrinkage estimator for dispersion improves differential expression detection in rna-seq data. *Biostatistics*, 14(2):232–243.

Xu, Z. and Taylor, J. (2009). Snpinfo: integrating gwas and candidate gene information into functional snp selection for genetic association studies. *Nucleic Acids Res*, 37(Web server issue):W600–5.

Yeager, M., Orr, N., Hayes, R., Jacobs, K., Kraft, P., Wacholder, S., Minichiello, M., Fearnhead, P., Yu, K., Chatterjee, N., Wang, Z., Welch, R., Staats, B., Calle, E., Feigelson, H., Thun, M., Rodriguez, C., Albanes, D., Virtamo, J., Weinstein, S., Schumacher, F., Giovannucci, E., Willett, W., Cancel-Tassin, G., Cussenot, O., Valeri, A., Andriole, G., Gelmann, E., Tucker, M., Gerhard, D., Fraumeni, J. J., Hoover, R., Hunter, D., Chanock, S., and Thomas, G. (2007). Genome-wide association study of prostate cancer identifies a second risk locus at 8q24. *Nature Genetics*, 39(5):645–649.

Zhang, Y. (2008). Poisson approximation for significance in genome-wide ChIP-chip tiling arrays. *Bioinformatics*, 24(24):2825–2831.

Zhou, Y.-H., Barry, W., and Wright, F. (2013). Empirical pathway analysis, without permutation. *Biostatistics*, 14(3):573–585.

# A Appendix of Chapter 2

## A.1 Clopper and Pearson (1934) confidence intervals

The particular choice of the function $f$ used in Example 2.2 and the empirical studies in Section 2.4 computes "exact" Clopper and Pearson (1934) confidence intervals. We choose $f$ in such a way as to guarantee a joint coverage probability of $1 - \epsilon$ for all confidence intervals over all iterations, where the overall error probability $\epsilon$ is chosen by the user.

A sequence $(\eta_k)_{k \in \mathbb{N}_0}$ satisfying $\eta_0 = 0$ and $\eta_k \to \epsilon$ as $k \to \infty$ is used to control how $\epsilon$ is spent over the iterations of the algorithm. We will call $(\eta_k)$ `spending sequence`. Throughout Chapter 2 we use $\eta_k := \frac{k}{k+r}\epsilon$ for some constants $r > 0$ and $\epsilon > 0$ (all the parameters used for the simulation studies are given in Section 2.4.1).

We then define $f(S, k, \Delta)$ to be the Clopper and Pearson (1934) confidence interval based on $S$ and $k$ (see Algorithm 2.1) with a coverage probability of $1 - (\eta_k - \eta_{k-\Delta})/m$.

Precisely,

$$f(S,k,\Delta) := \begin{cases} [1 - q^{\text{Beta}}_{k-S,S+1}(\rho_k), 1 - q^{\text{Beta}}_{k+1-S,S}(1-\rho_k)] & 0 < S < k, \\ [0, 1 - \rho_k^{1/k}] & S = 0, \\ [\rho_k^{1/k}, 1] & S = k, \end{cases}$$

where $\rho_k = (\eta_k - \eta_{k-\Delta})/(2m)$. The quantiles $q^{\text{Beta}}_{\alpha,\beta}(\epsilon)$ of the Beta$(\alpha, \beta)$ distribution being used are defined by $\mathbb{P}(Z \leq q^{\text{Beta}}_{\alpha,\beta}(\epsilon)) = \epsilon$ for a random variable $Z$ with probability density function $\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} z^{\alpha-1}(1-z)^{\beta-1}$. The Clopper and Pearson (1934) confidence intervals we compute are slightly conservative in practice (Li et al., 2009).

We show in the following lemma that our particular choice of $f$ defined beforehand satisfies Condition 2.5 and Condition 2.7 stated in Section 2.2.2. Therefore, $\epsilon$ is a bound on the probability of having any false classification. Other functions $f$, for example based on other spending sequences, can obviously be used as long as they satisfy Conditions 2.5 and 2.7.

**Lemma A.1.** *The confidence intervals computed by the function $f$ satisfy Conditions 2.5 and 2.7.*

*Proof.* First, we consider an individual Clopper and Pearson (1934) confidence interval $I_i^n$ computed in Algorithm 2.1 using $f$ as defined in Section A.1. To ease notation, we drop the indices $i$ and $n$.

We show that $|I| \leq 2\xi$, where $\xi = \sqrt{\frac{-1}{2k} \log \rho}$ and $\rho = (\eta_k - \eta_{k-\Delta})/(2m)$. The following probabilities are conditional on $S$ and $k$.

Suppose $S < k$. Then the upper limit $p_u$ of the interval $I$ is the solution to $\mathbb{P}(N \leq S | p = p_u) = \rho$, where $N \sim$ Binomial$(k, p)$. If $p > S/k + \xi$ then, by Hoeffding's inequality (Hoeffding, 1963),

$$\mathbb{P}(N \leq S) = \mathbb{P}\left(\frac{N}{k} - \mathbb{E}\left(\frac{N}{k}\right) \leq \frac{S}{k} - \mathbb{E}\left(\frac{N}{k}\right)\right) \leq \exp\left(-\frac{2(S/k-p)^2k^2}{k}\right) < \rho.$$

Thus $p_u \leq S/k + \xi$. If $S = k$ then $p_u = 1$, implying $p_u \leq S/k + \xi$.

Similarly, it can be shown that the lower limit $p_l$ of $I$ satisfies $p_l \geq S/k - \xi$. Hence, $|I| = p_u - p_l \leq 2\xi$.

Now consider $|f(\cdot, k, \cdot)|$ for $k \to \infty$, see Condition 2.5. The function $f(S, k, \Delta)$ given in Section A.1 computes Clopper and Pearson (1934) confidence intervals with coverage probability $1 - (\eta_k - \eta_{k-\Delta})/m$, where $\eta_k := \frac{k}{k+r}\epsilon$ for a constant $r > 0$. The sequence $\eta_k$ satisfies $\eta_k - \eta_{k-\Delta} \sim k^{-2}$, implying $\log(\eta_k - \eta_{k-\Delta}) = o(k)$. As $|I_i^n| \leq 2\sqrt{\log((\eta_k - \eta_{k-\Delta})/(2m))/(-2k)}$, $|I_i^n| \to 0$ as $k \to \infty$. This proves Condition 2.5.

Second, we show that the function $f$ given in Section A.1 computes confidence intervals in such a way that $\mathbb{P}(p_i^* \in I_i^n \; \forall i, n) \geq 1 - \epsilon$, thus satisfying Condition 2.7.

Let $k_i^n$ denote the value of $k_i$ in iteration $n$, and let $\Delta^n$ denote the value of $\Delta$ in iteration $n$, where $k_i^0 = k_i^1 - \Delta^1 = 0$, $i \in \{1, \ldots, m\}$. The function $f$ defined in Section A.1 computes Clopper and Pearson (1934) confidence intervals $I_i^n$ such that $\mathbb{P}(p_i^* \notin I_i^n) \leq (\eta_{k_i^n} - \eta_{k_i^n - \Delta^n})/m$.

This then yields

$$\mathbb{P}(\exists i, n : p_i^* \notin I_i^n) \leq \sum_{i=1}^{m}\sum_{n=1}^{\infty}\mathbb{P}(p_i^* \notin I_i^n) \leq \sum_{i=1}^{m}\sum_{n=1}^{\infty}(\eta_{k_i^n} - \eta_{k_i^n - \Delta^n})/m = \frac{1}{m}\sum_{i=1}^{m}\epsilon = \epsilon,$$

using properties of $\eta_k = \frac{k}{k+r}\epsilon$, where $r > 0$ is constant. Condition 2.7 is thus satisfied.

□

## A.2 The SAM statistic

In Section 2.4 we analyse a yeast chemostat cultivation dataset of Knijnenburg et al. (2009). This dataset consists of 170 microarrays of yeast cultivations. The first 80 microarrays correspond to yeast which was grown aerobically, the second 90 microarrays correspond to yeast which was grown anaerobically. Every microarray reacts to 9335 genes, thus giving rise to a multiple hypothesis problem with 9335 hypotheses.

We want to detect the differential expression of genes between the two types of yeast using a permutation test for each gene.

We use the test statistic SAM (Significance Analysis of Microarrays) of Tusher et al. (2001) as the basis for the permutation test. The SAM statistic works as follows: Let a permutation of the data be given which divides the expression data $x(i)$ of a single gene into two groups $I$ of size $l_1 = 80$ and $U$ of size $l_2 = 90$. Then, the *relative difference* $d(i)$ in expression for gene $i$ is given by

$$d(i) = \frac{\overline{x}_I(i) - \overline{x}_U(i)}{s(i) + s_0},$$

where $\overline{x}_I(i)$ is the average in gene expression for gene $i$ in group $I$ and $\overline{x}_U(i)$ is the average in gene expression for gene $i$ in group $U$. The so-called *gene-specific scatter* $s(i)$ is defined as

$$s(i) = \sqrt{a\left(\sum_m (x_m(i) - \overline{x}_I(i))^2 + \sum_n (x_n(i) - \overline{x}_U(i))^2\right)},$$

where $\sum_m$ denotes the sum over all expressions in group $I$ and $\sum_n$ denotes the sum over all expressions in group $U$. The constant $a$ is defined as $a = (1/l_1 + 1/l_2)/(l_1 +$

$l_2 - 2)$, where $l_1$ and $l_2$ are the group sizes of group $I$ and $U$, respectively.

As suggested in Tusher et al. (2001), a constant $s_0$ is added to the denominator of $d(i)$. The rationale for this is as follows: SAM compares values of $d(i)$ across genes having expressions with different means and variances. Especially for low expression levels, the authors note in Tusher et al. (2001) that the variance in $d(i)$ can be high. To resolve this problem, the normalising constant $s_0$ is added to the denominator of the SAM statistic. The value of $s_0$ is determined by fitting the statistic to the data in its initial partioning into two classes $I$ and $U$: $s_0$ is chosen in such a way as to minimise the coefficient of variation of $d(i)$ across all genes and is then kept fixed for all permutations. We used the $R$ package `samr` to fit $s_0$ to the yeast cultivation dataset of Knijnenburg et al. (2009).

## A.3 The permutation test p-value

As it is usually the case in practice, listing all $\binom{l}{l_1} = l!/(l_1!(l-l_1)!)$ permutations of an observation of length $l$ (divided into two groups of sizes $l_1$ and $l_2 = l - l_1$) is infeasible. This is also the case for the dataset of Knijnenburg et al. (2009) consisting of $l = 170$ microarrays. The permutation test p-value is thus estimated using a number $\lambda \leq \binom{l}{l_1}$ of samples.

For each gene $i$, let the $\lambda$ permutations of its expression data be denoted by $d(i)^*_j$ for $j \in \{1, \dots, \lambda\}$. We approximate the permutation test p-value as done in Knijnenburg et al. (2009) by comparing the permutation values $|d(i)^*_j|$ for all the permutations $j \in \{1, \dots, \lambda\}$ directly to the value of $|d(i)|$ for the reference statistic. As proposed in Davison and Hinkley (1997), a pseudocount is usually added in both

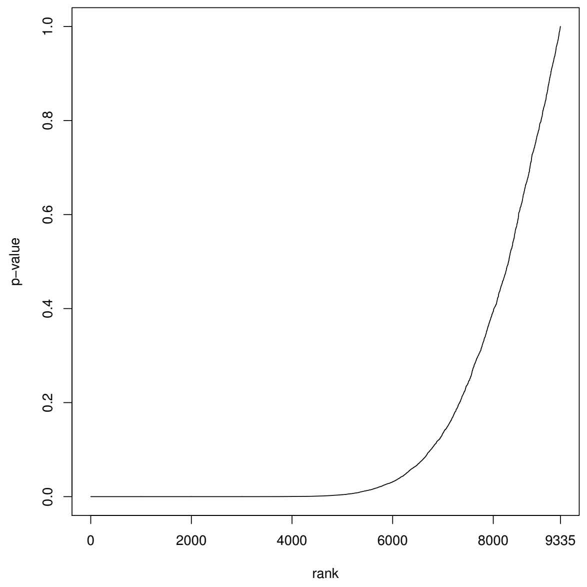


Figure A.1: Fixed p-values for the Knijnenburg et al. (2009) dataset.

the numerator and denominator. The permutation test p-value we compute is thus

$$\hat{p}_{\text{perm}}(i) = \frac{\sum_{j=1}^{\lambda} \mathbb{I}(|d(i)_j^*| \geq |d(i)|) + 1}{\lambda + 1},$$

where $\mathbb{I}$ denotes the indicator function.

The permutation p-value we compute is obtained through sampling with replacement. For permutation p-values without replacement see Phipson and Smyth (2010).

## A.4 The dataset

In order to be able to speed up the computation of the simulation studies and in order to have an underlying "truth" for the real data study, we estimated each of the $m = 9335$ p-values for the Knijnenburg et al. (2009) dataset once by generating $10^6$ permutations per hypothesis as outlined in Section A.2 and Section A.3. Such

Table A.1: Comparison of the naive method to `MMCTest` for a high proportion of true null hypotheses

| naive method | | | | MMCTest | | |
|---|---|---|---|---|---|---|
| | | | | guaranteed classification | forced classification | |
| *s* | *mis* | *rc* | *N* | *unclassified hypotheses* | *mis* | *rc* |
| 100 | 368 | 0* | 5e+05 | 860 | 99 | 220 |
| 1000 | 30 | 94 | 5e+06 | 225 | 8 | 23 |
| 10000 | 7 | 23 | 5e+07 | 6 | 0.1 | 1 |

*s*: number of samples used by the naive method for each hypothesis; *mis*: average number of misclassifications; *rc*: number of randomly classified hypotheses; *N*: average total number of samples.

a large number of permutations is far more than what would commonly be used in practice. We then define these approximated p-values to be the p-values $p_1^*, \dots, p_m^*$ we are interested in, although they do not necessarily have to be equal to the p-values underlying each hypothesis. A plot of the fixed p-values is given in Figure A.1.

As done in the main chapter, we draw Bernoulli samples with success probabilities $p_1^*, \dots, p_m^*$ in order to obtain new samples instead of generating actual permutations. The classification obtained by applying the Benjamini-Hochberg procedure directly to the p-values $p_1^*, \dots, p_m^*$ is used to compute misclassifications.

## A.5 Comparison of `MMCTest` to the naive method and to `MCFDR` on a dataset with a high proportion of hypotheses from the null

We conducted another comparison of `MMCTest` to the naive method and to `MCFDR` using a different distribution. The p-value distribution of Knijnenburg et al. (2009) used in Chapter 2 has a low proportion of true null hypotheses, whereas the one used in this section has a large proportion of true null hypotheses.

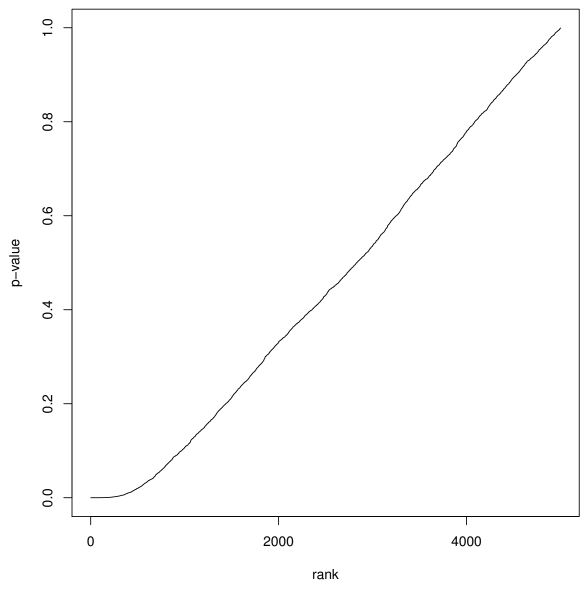


Figure A.2: Model of a realistic distribution with a proportion $\pi_0 = 0.9$ of true null hypotheses.

We used the setup introduced in Sandve et al. (2011) to model p-value distributions for gene expression data for a given proportion $\pi_0$ of true null hypotheses. Sandve et al. (2011) generate $m$ p-values from a mixture distribution consisting of a proportion $\pi_0$ of p-values drawn from a Uniform$[0, 1]$ distribution and a proportion $1 - \pi_0$ of p-values drawn from a Beta$(0.25, 25)$ distribution.

In this section, we analyse a simulated dataset generated from the mixture distribution described beforehand with $m = 5000$ hypotheses and a proportion $\pi_0 = 0.9$ of hypotheses from the null. A plot of this distribution is given in Figure A.2.

The setup of Sandve et al. (2011) allows to generate p-value distributions which qualitatively resemble the ones occurring in real data studies (compare Figure A.2 showing the p-values we use in this section to Figure A.1) while allowing to select the number of hypotheses from the null.

Apart from the new dataset, the setup we use in this section is the same as the one used in the simulation studies of Chapter 2. All the studies presented in this section are based on 1000 repetitions.

Table A.1 confirms the overall picture already seen in Section 2.4.3. The classifications of the naive method generally contain a large number of misclassifications and randomly classified hypotheses. On a realistic dataset, 23 random classifications occur on average (out of $m = 5000$ hypotheses), even at a high precision of $s = 10000$ samples.

No randomly classified hypotheses occur in the value annotated with a star in Table A.1 because of the following. When using a pseudocount as proposed in Davison and Hinkley (1997) (see Section A.3), the low resolution of only $s = 100$ replicates per p-value estimate causes all hypotheses to be consistently non-rejected in all the runs, independently of the actual p-value estimates. Thus, no randomly classified hypotheses occur. This phenomenon is identical to the one already observed in Section 4.2.2.

`MMCTest` was run on the same p-values using at most the overall number of samples spent by the naive method. `MMCTest` is not capable of classifying many hypotheses with confidence when only using $s = 100$ samples per hypotheses (i.e. a total number of $5 \cdot 10^5$ samples): around 860 unclassified hypotheses remain on average after each run. However, at a high precision ($s = 10000$ samples per hypothesis), `MMCTest` is able to classify all hypotheses with confidence except for a number of hypotheses which is smaller than the number of misclassifications incurred by the naive method.

As stressed in Chapter 2, all the classifications of `MMCTest` except for the unclassified hypotheses are correct up to the pre-specified error probability while the naive method does not give any guarantee on its result.

Table A.2: Comparison of `MCFDR` to `MMCTest` for a high proportion of true null hypotheses

| MCFDR | | | | MMCTest | | |
|---|---|---|---|---|---|---|
| | | | | guaranteed classification | forced classification | |
| *u* | *mis* | *rc* | *N* | *unclassified hypotheses* | *mis* | *rc* |
| 10 | 24 | 81 | $1.1 \cdot 10^6$ | 605 | 28 | 91 |
| 20 | 16 | 55 | $1.8 \cdot 10^6$ | 537 | 19 | 65 |
| 50 | 9 | 32 | $4.1 \cdot 10^6$ | 300 | 9 | 31 |
| 100 | 6 | 18 | $8.1 \cdot 10^6$ | 118 | 4 | 12 |
| 200 | 4 | 12 | $16.1 \cdot 10^6$ | 40 | 1.4 | 6 |
| 500 | 2.2 | 6 | $40.2 \cdot 10^6$ | 7 | 0.3 | 1 |
| 1000 | 1.4 | 6 | $80.3 \cdot 10^6$ | 3 | 0.03 | 1 |

$u$: number of test statistics exceeding the reference statistic (tuning parameter of `MCFDR`); *mis*: average number of misclassifications; *rc*: number of randomly classified hypotheses; $N$: average total number of samples.

The forced classification method of `MMCTest` based on point estimates yields considerably less misclassifications and random classifications than the naive method, especially at a high precision.

The comparison to the `MCFDR` algorithm of Sandve et al. (2011) in Table A.2 confirms these results. `MCFDR` yields less misclassifications and randomly classified hypotheses as the precision increases, but still incurs around 6 randomly classified hypotheses (on average) at a high precision.

As noted before, `MMCTest` yields a large number of unclassified hypotheses at a low precision. At a high precision, using the same number of samples as `MCFDR`, `MMCTest` classifies all hypotheses except for 3 (out of $m = 5000$ hypotheses). This almost equals the number of misclassifications of `MCFDR`, for which no guarantee is given, and it is less than the number of random classifications of `MCFDR` at a high precision.

The forced classification method of `MMCTest` quickly overtakes `MCFDR` in terms of misclassifications and randomly classified hypotheses as the precision increases. As already seen in Table A.1 for the naive method, `MMCTest` computes classifications at a high precision which are virtually free of misclassifications and only contain one random classification on average.

# B Appendix of Chapter 3

## B.1 Proofs

For simplicity of notation, the dependence of the multiple testing procedure $h(p, \alpha)$ on the threshold $\alpha$ is sometimes omitted.

The following two lemmas will be needed for the proof of Lemma 3.12. First, Lemma B.1 proves three properties of step-up and step-down procedures which are slightly stronger than the requirements stated in Condition 3.5. For a vector $p = (p_1, \dots, p_m)$, we denote the rank of $p_i$ in the sorted sequence $p_{(1)} \leq \dots \leq p_{(m)}$ by $r_p(i)$. Lemma B.1 generalises Lemma 2.9 for arbitrary step-up and step-down procedures.

**Lemma B.1.** *Let $p, q \in [0,1]^m$. Let $h_u$ ($h_d$) be a step-up (step-down) procedure defined through a threshold function $\tau_\alpha$ satisfying Condition 3.11.*

1. *$h_u$ is monotonic.*
2. *If $q_i \leq \tau_\alpha(|h_u(p)|)$ $\forall i \in h_u(p)$ and $q_i = p_i$ $\forall i \notin h_u(p)$, then $h_u(p) = h_u(q)$.*
3. *If $q_i = p_i$ $\forall i \in h_u(p)$ and $q_i > \tau_\alpha(r_p(i))$ $\forall i \notin h_u(p)$, then $h_u(p) = h_u(q)$.*
4. *$h_d$ is monotonic.*
5. *If $q_i \leq \tau_\alpha(r_p(i))$ $\forall i \in h_d(p)$ and $q_i = p_i$ $\forall i \notin h_d(p)$, then $h_d(p) = h_d(q)$.*

*6. If $q_i = p_i \ \forall i \in h_d(p)$ and $q_i > \tau_\alpha(|h_d(p)|+1) \ \forall i \notin h_d(p)$, then $h_d(p) = h_d(q)$.*

*Proof.* As $h_u$ and $h_d$ are invariant to permutations, we may assume $p_1 \leq \cdots \leq p_m$.

1. Let $p \in [0,1]^m$ and $i \in \{1,\dots,m\}$. It suffices to show that $h_u(p) \supseteq h_u(q)$ for any $q \in [0,1]^m$ given by $q_j = p_j \ \forall j \neq i$ and $q_i > p_i$.

Let $k := |h_u(p)|$ be the largest rejected index. We need to show that $j \notin h_u(q)$ $\forall j \geq k+1$. Let $\alpha$ be fixed.

Case 1: $r_q(i) \leq k$. This implies $r_q(j) = j \ \forall j \geq k+1$ and hence $q_j = p_j > \tau_\alpha(j) = \tau_\alpha(r_q(j))$. Therefore, $j \notin h_u(q) \ \forall j \geq k+1$.

Case 2: $r_q(i) \geq k+1$. Let $j \geq k+1$, $j \neq i$. Then the rank of the $j$th p-value can only drop by one when $p_i$ is replaced by $q_i$, i.e. $r_q(j) \in \{j-1, j\}$. Thus $q_j = p_j > \tau_\alpha(j) \geq \tau_\alpha(r_q(j))$ on Condition 3.11 (using that $\tau_\alpha(i)$ is non-decreasing in $i$). Furthermore, as $r_q(i) \geq k+1$, $q_i$ takes the position of the former $p_{r_q(i)}$ in the ordered sequence of values from $q$, i.e. $q_i \geq p_{r_q(i)}$. Hence, $r_q(i) \notin h_u(p)$ because of $r_q(i) \geq k+1$ and thus $q_i \geq p_{r_q(i)} > \tau_\alpha(r_q(i))$. Therefore, $\{k+1,\dots,m\} \cup \{i\} \notin h_u(q)$. This proves the monotonicity in the first argument of $h_u$.

The monotonicity in the second argument of $h$ is immediate as $p_i \leq \max_j\{p_{(j)} : p_{(j)} \leq \tau_\alpha(j)\}$ for all $i \in h_u(p,\alpha)$. On Condition 3.11, using that $\tau_\alpha$ is non-decreasing in $\alpha$, $\alpha \leq \alpha'$ implies $\tau_\alpha(j) \leq \tau_{\alpha'}(j) \ \forall j$, hence $i \in h_u(p,\alpha')$. This proves 1.

2. All $i \notin h_u(p)$ satisfy $p_i > \tau_\alpha(r_p(i)) > \tau_\alpha(|h_u(p)|)$ whereas by assumption, $q_i \leq \tau_\alpha(|h_u(p)|) \ \forall i \in h_u(p)$. Hence, using $q_i = p_i \ \forall i \notin h_u(p)$, it follows that $r_q(i) = r_p(i) \ \forall i \notin h_u(p)$. Thus, $q_i = p_i > \tau_\alpha(r_p(i)) = \tau_\alpha(r_q(i))$ for all $i \notin h_u(p)$. Hence $h_u(p)^c \subseteq h_u(q)^c$.

Conversely, define $\tilde{q} := \max\{q_i : i \in h_u(p)\}$. As $\tilde{q} \leq \tau_\alpha(|h_u(p)|) < q_i$ for all $i \notin h_u(p)$ and as there are precisely $|h_u(p)|$ values $q_i \leq \tilde{q}$, the rank of $\tilde{q}$ in $q$ is precisely $|h_u(p)|$. As $q_i \leq \tilde{q} \leq \tau_\alpha(|h_u(p)|)$ $\forall i \in h_u(p)$, all $\{q_i\}_{i \in h_u(p)}$ are rejected, so $h_u(p) \subseteq h_u(q)$. This proves 2.

3. As $q_i = p_i$ for all $i \in h_u(p)$, we have $h_u(p) \subseteq h_u(q)$.

Let $i \notin h_u(p)$. If $r_q(i) \leq r_p(i)$, then $q_i > \tau_\alpha(r_p(i)) \geq \tau_\alpha(r_q(i))$ by Condition 3.11. If $r_q(i) > r_p(i)$, $q_i$ replaces a $q_j > \tau_\alpha(r_p(j))$ at rank $r_p(j)$ in the sorted sequence of $q$, hence $r_q(i) = r_p(j)$ and $q_i \geq q_j > \tau_\alpha(r_p(j)) = \tau_\alpha(r_q(i))$. Thus $q_i > \tau_\alpha(r_q(i))$ $\forall i \notin h_u(p)$, which implies $h_u(p)^c \subseteq h_u(q)^c$. This proves 3.

In a similar fashion, 4., 5. and 6. can be proven for step-down procedures $h_d$. □

For step-up procedures $h_u$, part 2. of Lemma B.1 shows that p-values of rejected hypotheses can be increased up to $\tau_\alpha(|h_u(p)|)$, the threshold evaluated at the last rejected hypothesis, without affecting the result of $h_u$. Part 3. of Lemma B.1 shows that $h_u$ is invariant if p-values in the non-rejection area are replaced by arbitrary values above the threshold (at their ranks).

Similarly, step-down procedures $h_d$ are invariant if p-values of rejected hypotheses are replaced by arbitrary values below the threshold (part 5.) or p-values of non-rejected hypotheses are replaced by arbitrary values above $\tau_\alpha(|h_d(p)|+1)$, the threshold evaluated at the first non-rejected hypothesis (part 6.).

The following Lemma B.2 will also be needed for the proof of Lemma 3.12. In the following, $\|\tau_\alpha\|_\infty$ shall denote the maximal value attained by $\tau_\alpha : \{1,\dots,m\} \to [0,1]$ on $\{1,\dots,m\}$.

Lemma B.2 generalises Lemma 2.11 proven for the Benjamini and Hochberg (1995) procedure: it provides the same statement, though now for the threshold function of an arbitrary step-up and step-down procedure and it does not require the testing threshold to be constant any more.

**Lemma B.2.** *Let $h$ stand for $h_u$ or $h_d$. If $p^* \in [0,1]^m$, $\alpha^* > 0$ with $p^*_{(i)} \neq \tau_{\alpha^*}(i)$ $\forall i \in \{1,\dots,m\}$, then there exists $\delta > 0$ such that $p \in [0,1]^m$, $\tau_\alpha : \{1,\dots,m\} \to [0,1]$ and $\|p^* - p\| \vee \|\tau_{\alpha^*} - \tau_\alpha\|_\infty < \delta$ $\forall i \in \{1,\dots,m\}$ imply $h(p,\alpha) = h(p^*,\alpha^*)$.*

*Proof.* Let

$$\delta' = \min\left(\left\{\frac{p^*_i - p^*_{i-1}}{2} : p^*_{i-1} < p^*_i\right\}_{i=1,\dots,m} \cup \{|p^*_i - \tau_{\alpha^*}(i)|\}_{i=1,\dots,m}\right)$$

and let $\delta = \delta'/2$.

By assumption, $\|p - p^*\| < \delta < \delta'$, hence $p_{i-1} < p^*_{i-1} + \delta' \leq p^*_i - \delta' < p_i$. This means that $p^*_i$ and $p_i$ have the same ranks in $p^*$ and $p$, respectively.

Moreover, $|p^*_i - \tau_{\alpha^*}(i)| \leq |p^*_i - \tau_\alpha(i)| + |\tau_\alpha(i) - \tau_{\alpha^*}(i)| \leq |p^*_i - \tau_\alpha(i)| + \|\tau_\alpha - \tau_{\alpha^*}\|_\infty \leq |p^*_i - \tau_\alpha(i)| + \delta$. Hence $2\delta = \delta' \leq |p^*_i - \tau_{\alpha^*}(i)| \leq |p^*_i - \tau_\alpha(i)| + \delta$, meaning that $\delta \leq |p^*_i - \tau_\alpha(i)|$ for $i \in \{1,\dots,m\}$.

So $|p^*_i - p_i| < \delta \leq |p^*_i - \tau_\alpha(i)|$, hence $p_i$ and $p^*_i$ lie on the same side of the testing threshold. □

*Proof of Lemma 3.12.* 1. The monotonicity of $h_u$ and $h_d$ follows from Lemma B.1 (part 1.) and (part 4.), respectively.

2. To prove that $h_u$ satisfies the first part of Condition 3.5, it suffices to show that for $p, q \in [0,1]^m$, both $q_i \leq p_i$ $\forall i \in h_u(p)$ and $q_i = p_i$ $\forall i \notin h_u(p)$ as well as $q_i = p_i$ $\forall i \in h_u(p)$ and $q_i \geq p_i$ $\forall i \notin h_u(p)$ imply $h_u(p) = h_u(q)$.

Indeed, let $p, q \in [0,1]^m$ be such that $q_i \leq p_i$ $\forall i \in h_u(p)$ and $q_i = p_i$ $\forall i \notin h_u(p)$. We have $p_i \leq \tau_\alpha(|h_u(p)|)$ $\forall i \in h_u(p)$, thus $q_i \leq p_i \leq \tau_\alpha(|h_u(p)|)$ $\forall i \in h_u(p)$ and $h_u(p) = h_u(q)$ by Lemma B.1 (part 2.).

Similarly, let $p, q \in [0,1]^m$ be such that $q_i = p_i$ $\forall i \in h_u(p)$ and $q_i \geq p_i$ $\forall i \notin h_u(p)$.

Using $p_i > \tau_\alpha(r_p(i))$ $\forall i \notin h_u(p)$, it instantly follows that $q_i \geq p_i > \tau_\alpha(r_p(i))$ $\forall i \notin h_u(p)$ and thus $h_u(p) = h_u(q)$ by Lemma B.1 (part 3.).

To prove that $h_d$ satisfies the first part of Condition 3.5, it equally suffices to show that for $p, q \in [0,1]^m$, both $q_i \leq p_i$ $\forall i \in h_d(p)$ and $q_i = p_i$ $\forall i \notin h_d(p)$ as well as $q_i = p_i$ $\forall i \in h_d(p)$ and $q_i \geq p_i$ $\forall i \notin h_d(p)$ imply $h_d(p) = h_d(q)$.

Indeed, let $p, q \in [0,1]^m$ be such that $q_i \leq p_i$ $\forall i \in h_d(p)$ and $q_i = p_i$ $\forall i \notin h_d(p)$. Using $p_i \leq \tau_\alpha(r_p(i))$ $\forall i \in h_d(p)$, it immediately follows that $q_i \leq p_i \leq \tau_\alpha(r_p(i))$ $\forall i \in h_d(p)$ and thus $h_d(p) = h_d(q)$ by Lemma B.1 (part 5.).

Similarly, let $p, q \in [0,1]^m$ be such that $q_i = p_i$ $\forall i \in h_d(p)$ and $q_i \geq p_i$ $\forall i \notin h_d(p)$. We have $p_i > \tau_\alpha(|h_d(p)| + 1)$ $\forall i \notin h_d(p)$, thus $q_i \geq p_i > \tau_\alpha(|h_d(p)| + 1)$ $\forall i \notin h_d(p)$ and $h_d(p) = h_d(q)$ by Lemma B.1 (part 6.).

3. As $\tau_\alpha(i)$ is continuous in $\alpha$ $\forall i \in \{1, \ldots, m\}$ by Condition 3.11, for each $\epsilon_i > 0$ there exists a $\delta_i > 0$ such that $|\alpha^* - \alpha| < \delta_i$ implies $|\tau_{\alpha^*}(i) - \tau_\alpha(i)| < \epsilon_i$. Applying continuity to $\epsilon_i = \delta$ yields a $\delta_i$ for each $i \in \{1, \ldots, m\}$, where $\delta > 0$ is given by Lemma B.2. The second part of Condition 3.5 then follows for all $p \in [0,1]^m$ and $\alpha \in [0,1]$ satisfying $\|p - p^*\| \vee |\alpha - \alpha^*| < \min\{\delta, \delta_1, \ldots, \delta_m\}$. □

## B.2 The Hommel procedure is not admissible

The Hommel (1988) procedure determines the largest index $k$ satisfying $p_{(m-k+j)} > j\alpha/k$ for all $j = 1, \ldots, k$ and then rejects all the $H_{0i}$ with $p_i \leq \alpha/k$. If no such $k$ exists, all hypotheses are rejected.

The Hommel (1988) procedure $h(p, \alpha)$ is not a classical step-up or step-down procedure. Given $p$, determining the index $k$ corresponds to applying $m$ step-up procedures $h_j$, $j \in \{1, \ldots, m\}$, to $P_j = (p_{(m-j+1)}, \ldots, p_{(m)})$ using the threshold functions $\tau_j(i) = i\alpha/j$, $i \in \{1, \dots, j\}$ (these step-up procedures are admissible). Once $k_p = \max\{j : h_j(P_j) = \emptyset\}$ is determined, rejections are calculated by applying the Bonferroni (1936) correction (defined in Section 3.4.2) at threshold $\alpha/k_p$ to all p-values $p$, i.e. $h(p, \alpha) = h_{\text{Bonferroni}}(p, m\alpha/k_p)$.

The Hommel (1988) procedure is monotonic (proven below in Section B.2.1 for completeness) but fails to satisfy Condition 3.5.

Indeed, for $q_i \geq p_i \ \forall i \notin h(p, \alpha)$, the first part of Condition 3.5 is not satisfied. Consider $p = [\alpha/3+\epsilon, \alpha/2+\epsilon, 1]$, where $0 < \alpha < 1$ and $0 < \epsilon \leq \alpha/6$. Then $h_1(P_1) = \emptyset$, $h_2(P_2) = \emptyset$, $h_3(P_3) = \{1, 2\}$, so $k_p = 2$. Therefore, $h(p, \alpha) = \{1\}$. Increasing $p_2$ to $p_2 = 2\alpha/3 + \epsilon$ yields $h_1(P_1) = h_2(P_2) = h_3(P_3) = \emptyset$, hence $k_p = 3$ and $h(p, \alpha) = \emptyset$.

### B.2.1 The Hommel procedure is monotonic

For $p = (p_1, \dots, p_m)$ and $j \in \{1, \dots, m\}$, let $P_j = (p_{(m-j+1)}, \dots, p_{(m)})$ as before and likewise define $Q_j = (q_{(m-j+1)}, \dots, q_{(m)})$ for $q = (q_1, \dots, q_m)$.

The Hommel (1988) procedure is monotonic in the first argument. Indeed, for $p \leq q$, $P_j \subseteq Q_j \ \forall j$ and thus $h_j(P_j) \supseteq h_j(Q_j)$ by monotonicity of $h_j$ (using that all $h_j$ are admissible). Therefore, $h_j(P_j) = \emptyset$ implies $h_j(Q_j) = \emptyset$, meaning that $k_p \leq k_q$. Using the monotonicity of the Bonferroni (1936) correction, $\alpha/k_p \geq \alpha/k_q$ implies $h(p, \alpha) \supseteq h(q, \alpha)$.

The monotonicity of $h(p, \alpha)$ in the second argument follows directly from the monotonicity of the Bonferroni (1936) correction.

# C Appendix of Chapter 4

## C.1 Zero rejected hypotheses occur at low effort

The naive method draws a constant number of samples per hypothesis and computes p-value estimates $\hat{p}_i = (e_i + 1)/(s + 1)$ as proposed in Davison and Hinkley (1997), where $e_i$ denotes the number of exceedances observed for hypothesis $H_{0i}$ among $s$ samples.

Assuming full knowledge of all p-values, let $\tau_0$ be the proportion of non-rejected hypotheses observed when applying the multiple testing procedure to the p-values. In order to observe the correct number of rejections with approximated p-values, the $(1 - \tau_0)m$th ordered p-value estimate has to be the last one lying below the critical value at its rank in the sorted sequence of estimates.

For the Hochberg (1988) procedure applied at $\alpha^* = 0.1$, a constant number of 87 misclassifications were observed for the naive method at both low and high effort in Table 4.1 of Section 4.3.1. Indeed, applying the Hochberg (1988) procedure to the fixed p-values leads to 87 rejections and 4913 non-rejections, thus $\tau_0 = 4913/5000 = 0.9826$. The critical value of the Hochberg (1988) procedure at the 87th p-value is approximately $2 \cdot 10^{-5}$.

Estimated p-values with a pseudo-count are bounded from below by $L = 1/(s+1)$, where $s$ is the number of samples. Even for $s = 10000$ (high effort), $L > 2 \cdot 10^{-5}$ and thus all p-value estimates lie above the (correct) rejection area. Although theoretically it would be possible that another (possibly underestimated) p-value lies below a critical value again at a higher rank (as most critical values are non-decreasing), this rarely happens as hypotheses having higher ranked p-values are typically non-rejected, hence their p-values (and estimates) are expected to be (much) larger than their corresponding critical values. Thus in most cases, the fact that $L > 2 \cdot 10^{-5}$ actually leads to all hypotheses being consistently non-rejected in each run, independently of the actual number of exceedances observed. It is hence impossible to record any rejection.

This phenomenon disappears in three cases. First, using more samples $s$ to estimate each p-value will decrease the lower bound $L$. Alternatively, datasets with less hypotheses from the null and thus lower $\pi_0$ lead to a higher threshold at the $(1-\pi_0)m$th p-value. Third, estimating p-values without a pseudo-count leads to estimates which attain a value of zero and hence allow one to observe rejections for any number of samples $s$, though using such estimates does not guarantee appropriate error control (Section 4.2.1).

## C.2 Simulation study at a variable testing threshold

The simulation study comparing the naive approach to `QuickMMCTest` (Section 4.3.1) was also carried out at a variable testing threshold. Apart from the testing threshold, the set-up we used, in particular the fixed p-value distribution, is identical to the one of Section 4.3.1.

Table C.1: Common multiple testing procedures at the variable testing threshold of Pounds and Cheng (2006) – Average misclassification numbers (average numbers of erroneously rejected hypotheses in brackets) for the naive method at a low effort ($s = 1000$) and a high effort ($s = 10000$) compared to `QuickMMCTest` (Alg. 4.1).

| | low effort ($s$=1000) | | high effort ($s$=10000) | |
|---|---|---|---|---|
| procedure | naive | Alg. 4.1 | naive | Alg. 4.1 |
| Bonferroni (1936) | 90 (0) | 40.4 (2.5) | 90 (0) | 3.5 (1.4) |
| Simes (1986) | 30.7 (10.1) | 1.9 (0.9) | 8.9 (4.4) | 0 (0) |
| Hochberg (1988) | 90 (0) | 39.9 (2.5) | 90 (0) | 3.5 (1.6) |
| Benjamini and Hochberg (1995) | 30.4 (9.9) | 1.9 (0.8) | 9.1 (4.5) | 0 (0) |
| Benjamini and Yekutieli (2001) | 168 (0) | 14 (3.8) | 22.5 (7) | 1.7 (1.2) |
| Sidak (1967) | 94 (0) | 31 (2.4) | 94 (0) | 3.1 (0.5) |
| Holm (1979) | 91 (0) | 33.3 (3) | 91 (0) | 3.4 (1.6) |

We corrected the threshold using the estimate $\hat{\pi}_0(p) = \min\left(1, 2/m \sum_{i=1}^{m} p_i\right)$ of the proportion $\pi_0$ of true null hypotheses of Pounds and Cheng (2006), employed in various real data studies (Han and Dalal, 2012; Lu et al., 2011; Jupiter et al., 2010; Cheng, 2009). The corrected testing threshold is given by $\alpha(p^*) = \alpha^*/\hat{\pi}_0(p^*)$, where $\alpha^* = 0.1$ is an uncorrected threshold.

To apply the naive method to a variable testing threshold, one simply computes usual p-value estimates $\hat{p}$ with a pseudo-count and classifies all hypotheses using these estimates at the plug-in threshold $\alpha(\hat{p})$.

Simulation results are given in Table C.1. These agree with the ones presented in Section 4.3.1. As shown in the table, the naive method either observes no rejections or performs poorly at a low effort when being applied to common multiple testing procedures. As observed in Section 4.3.1, Algorithm 4.1 is able to compute meaningful results for all procedures at a low effort, nevertheless its results contain up to 40 misclassifications. Its test results are again especially accurate for the procedures of

Table C.2: Average misclassification numbers (average numbers of erroneously rejected hypotheses in brackets) for common methods compared to `QuickMMCTest` for the Bonferroni (1936) correction. Threshold of Pounds and Cheng (2006) at $\alpha^* = 0.1$.

| | low effort ($K = 1000m$) | high effort ($K = 10000m$) |
|---:|---:|---:|
| Naive method | 90 (0) | 90 (0) |
| Besag and Clifford (1991) | 90 (0) | 5.8 (1.5) |
| Guo and Peddada (2008) | 90 (0) | 6.1 (1.1) |
| Sandve et al. (2011) | 90 (0) | 20.1 (1.3) |
| Jiang and Salzman (2012) | 90 (0) | 17.4 (2.7) |
| Gandy and H. (2014) | 90 (0) | 8.8 (1.1) |
| `QuickMMCTest` | 40.2 (2.3) | 3.6 (1.4) |

$K$: total number of samples; $m$: number of hypotheses.

Simes (1986) and Benjamini and Hochberg (1995), and generally only contain few erroneously rejected hypotheses.

Similarly to Section 4.3.1, the naive method still fails to record any rejections for some procedures at a high effort. Algorithm 4.1 yields a multiple fold decrease in misclassifications compared to the naive method and only few erroneously rejected hypotheses.

## C.3 Common methods at a variable testing threshold

We use the p-values fixed in Section 4.3 and evaluate all algorithms considered in Section 4.3.2 using the Bonferroni (1936) procedure at the variable testing threshold of Pounds and Cheng (2006) with an uncorrected threshold of $\alpha^* = 0.1$ (same testing setting as in Section C.2).

The comparison of commonly used methods to `QuickMMCTest` (Algorithm 4.1) in Table C.2 confirms the picture already observed at a constant threshold (Section

Table C.3: Average misclassification numbers (average numbers of erroneously rejected hypotheses in brackets) for common methods compared to `QuickMMCTest` using the Benjamini and Hochberg (1995) procedure. Testing threshold of Pounds and Cheng (2006) at $\alpha^* = 0.1$.

| | low effort ($K = 1000m$) | high effort ($K = 10000m$) |
|---:|---:|---:|
| Naive method | 32.1 (9.6) | 9 (3.4) |
| Besag and Clifford (1991) | 18.4 (7.5) | 18.8 (7.5) |
| Guo and Peddada (2008) | 4.4 (2) | 0.2 (0.2) |
| Sandve et al. (2011) | 10.5 (4.2) | 2.7 (1.3) |
| Jiang and Salzman (2012) | 13.7 (5.1) | 3.6 (1.6) |
| Gandy and H. (2014) | 9.8 (3.9) | 0.6 (0.5) |
| `QuickMMCTest` | 2.4 (1.2) | 0.2 (0.1) |

$K$: total number of samples; $m$: number of hypotheses.

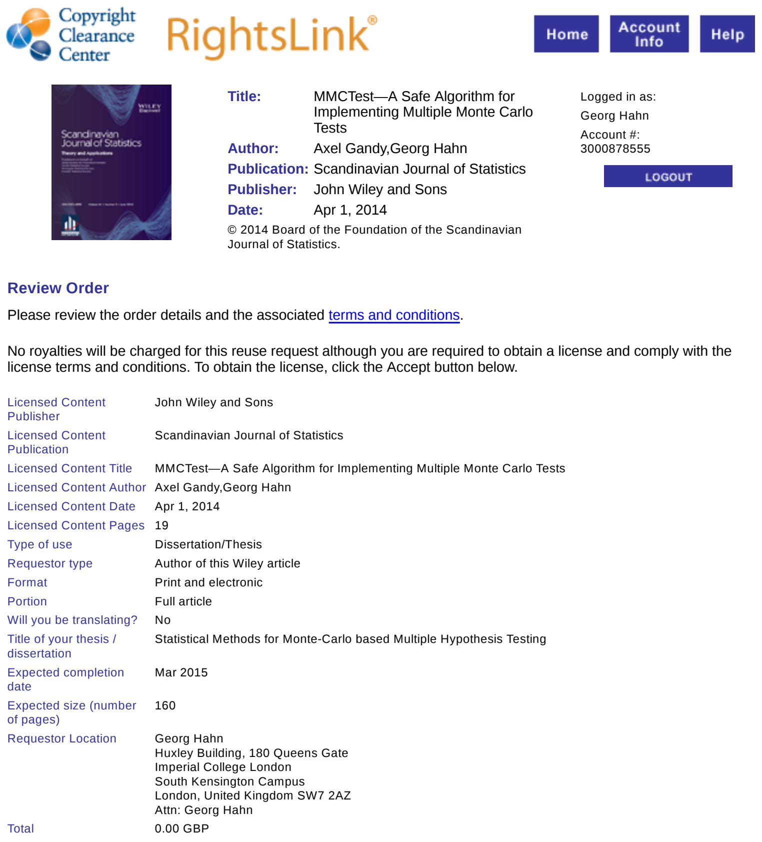
4.3.2): due to the low threshold, all methods except for Algorithm 4.1 fail to compute meaningful test results at a low effort.

At a high effort, the naive method is again unable to observe any rejections, whereas most other methods yield test results with reasonable accuracy. The methods of Besag and Clifford (1991) and Guo and Peddada (2008) perform especially well and are only outperformed by `QuickMMCTest`.

When using the Benjamini and Hochberg (1995) procedure, the picture observed in Table C.3 again resembles the one observed in Section 4.3.2: For the Benjamini and Hochberg (1995) procedure, all methods are able to observe rejections and thus to compute meaningful test results for both a low and a high effort. As in Section 4.3.2, the method of Guo and Peddada (2008) and `QuickMMCTest` perform comparably well and considerably better than the other methods. The precise numbers in Table C.3 are almost the same as the ones in Table 4.3 of Section 4.3.2.

# D Permissions to re-use own work

Permission for the article “MMCTest – A Safe Algorithm for Implementing Multiple Monte Carlo Tests” published in Gandy and H. (2014):



| | |
|---|---|
| **Title:** | MMCTest—A Safe Algorithm for Implementing Multiple Monte Carlo Tests |
| **Author:** | Axel Gandy,Georg Hahn |
| **Publication:** | Scandinavian Journal of Statistics |
| **Publisher:** | John Wiley and Sons |
| **Date:** | Apr 1, 2014 |

© 2014 Board of the Foundation of the Scandinavian Journal of Statistics.

Logged in as:
Georg Hahn
Account #:
3000878555

**Review Order**

Please review the order details and the associated terms and conditions.

No royalties will be charged for this reuse request although you are required to obtain a license and comply with the license terms and conditions. To obtain the license, click the Accept button below.

| | |
|---|---|
| Licensed Content Publisher | John Wiley and Sons |
| Licensed Content Publication | Scandinavian Journal of Statistics |
| Licensed Content Title | MMCTest—A Safe Algorithm for Implementing Multiple Monte Carlo Tests |
| Licensed Content Author | Axel Gandy,Georg Hahn |
| Licensed Content Date | Apr 1, 2014 |
| Licensed Content Pages | 19 |
| Type of use | Dissertation/Thesis |
| Requestor type | Author of this Wiley article |
| Format | Print and electronic |
| Portion | Full article |
| Will you be translating? | No |
| Title of your thesis / dissertation | Statistical Methods for Monte-Carlo based Multiple Hypothesis Testing |
| Expected completion date | Mar 2015 |
| Expected size (number of pages) | 160 |
| Requestor Location | Georg Hahn<br>Huxley Building, 180 Queens Gate<br>Imperial College London<br>South Kensington Campus<br>London, United Kingdom SW7 2AZ<br>Attn: Georg Hahn |
| Total | 0.00 GBP |

**JOHN WILEY AND SONS LICENSE**
**TERMS AND CONDITIONS**

Feb 05, 2015

This Agreement between Georg Hahn ("You") and John Wiley and Sons ("John Wiley and Sons") consists of your license details and the terms and conditions provided by John Wiley and Sons and Copyright Clearance Center.

| | |
|---|---|
| License Number | 3562420498476 |
| License date | Feb 05, 2015 |
| Licensed Content Publisher | John Wiley and Sons |
| Licensed Content Publication | Scandinavian Journal of Statistics |
| Licensed Content Title | MMCTest—A Safe Algorithm for Implementing Multiple Monte Carlo Tests |
| Licensed Content Author | Axel Gandy,Georg Hahn |
| Licensed Content Date | Apr 1, 2014 |
| Pages | 19 |
| Type of use | Dissertation/Thesis |
| Requestor type | Author of this Wiley article |
| Format | Print and electronic |
| Portion | Full article |
| Will you be translating? | No |
| Title of your thesis / dissertation | Statistical Methods for Monte-Carlo based Multiple Hypothesis Testing |
| Expected completion date | Mar 2015 |
| Expected size (number of pages) | 160 |
| Requestor Location | Georg Hahn<br>Huxley Building, 180 Queens Gate<br>Imperial College London<br>South Kensington Campus<br>London, United Kingdom SW7 2AZ<br>Attn: Georg Hahn |
| Billing Type | Invoice |
| Billing Address | Georg Hahn<br>Huxley Building, 180 Queens Gate<br>Imperial College London<br>South Kensington Campus<br>London, United Kingdom SW7 2AZ<br>Attn: Georg Hahn |
| Total | 0.00 GBP |
| Terms and Conditions | |

## TERMS AND CONDITIONS