Evaluating the robustness of permutation-based multiple testing methods

Kisoentewari, A.M.K.

Master Thesis Statistics & Data Science

Universiteit Leiden

Evaluating the robustness of permutation-based multiple testing methods

---

Author
A.M.K. (Anish) Kisoentewari

Advisor
dr. JBA (Jesse) Hemerik, Wageningen University & Research

Defended on 23-08, 2022

Abstract

Statistical hypothesis testing is central to many scientific fields. Testing many hypotheses simultaneously is called multiple testing. The main concern in multiple testing, is to ensure that most of the rejected null hypotheses are indeed false, i.e., that the number of incorrect rejections remains low. A major challenge in multiple testing is to account for the complex dependencies in the data. A powerful approach in this regard, are permutation-based multiple testing methods. These methods make few distributional assumptions. In fact, they often make only one assumption, called joint exchangeability. In this thesis we investigate the robustness of the methods to violations of this assumption. We do this by means of simulations, where we focus on case-control data. We find that, while the theoretical literature always makes the mentioned assumption, it is often not necessary in practice. Thus, this thesis provides further evidence for the validity of these powerful methods in practice.

Keywords: Multiple testing; Permutation; High-dimensional; maxT; SAM; Robustness; Heteroscedasticity

Foreword

First and foremost I would like to express my gratitude for the excellent guidance and supervision provided by dr. Hemerik. He was always available to answer any of my questions and provided me with constructive and critical feedback throughout the entire process. I would also like to thank Leiden University and Wageningen University & Research for providing me with the thesis opportunity, although the general idea of the thesis subject came from dr. Hemerik himself. Further decision making, writing and scripting was done by myself. No confidential data or restricted sources were used.

Contents

1 Introduction

In modern hypothesis testing, multiple testing settings arise more and more frequently. With the advent of big data sets, hundreds if not thousands of statistical inferences are to be made simultaneously. A common field of study where multiple testing problems occur is biomedical sciences. With the possibility of extraction of gene expressions from DNA, RNA and several other sources, genetic profiles containing information on a large number of genes may be constructed. This information may be used to examine associations between the extracted gene expressions and a measured phenotype (Menyhart, Weltz, & Győrffy, 2021). In order to present found associations as conclusive, some sort of hypothesis test is required. However, hypothesis testing is subject to errors, especially in the multiple testing setting.

The outcomes of hypothesis tests are displayed in Table 1. Two types of errors may be discerned. The first error is the type I error. This happens when one rejects a null hypothesis H_0, which happens to be true. This is also called a false discovery, or false positive. The acceptance rate of a false positive is expressed by the significance level (α), which is by convention often set to 0.05. Hence the control of the type I error may be managed by α. However, lowering α to prevent Type I errors indirectly causes fewer rejections, including fewer rejections for the group of false null hypotheses, hence lowering the power $(1-\beta)$ of a hypothesis test, and increasing the probability of a type II error (β), assuming there are false null hypotheses (Park, 2008).

Table 1: Hypothesis testing conclusions

	True H_0	**False H_0**
Reject H_0	Type I error ($\boldsymbol{\alpha}$)	Correct decision ($\mathbf{1-\beta}$)
Fail to reject H_0	Correct decision	Type II error ($\boldsymbol{\beta}$)

Mitigation of Type I errors is generally considered more important than mitigation of Type II errors. Failing to reject a false H_0 means a finding is overlooked, but may be found in another research or experiment. Rejecting a true H_0 means a novel finding is presented, which is not valid at all, possibly causing misconceptions or even contradicting scientific evidence (Goeman, 2017). For a relatively small amount of hypotheses being tested simultaneously, Type I error rate control through the use of α is historically considered sufficient by the scientific community. An expansion of the problem of Type I errors occurs when a large number of hypotheses need to be inferred simultaneously. Multiple Type I errors may now occur. We denote the total number of Type I errors among all inferences as V. In this case, among the count of all true hypotheses (m_0), the probability of at least one false discovery, under independence of the hypotheses, may be expressed as:

$$P(V > 0) = 1 - (1 - \alpha)^{m_0} \tag{1}$$

For only one true hypothesis, the probability of at least one false discovery reduces to α. However, for more true hypotheses, the probability of at least one type I error among all inferences increases drastically, as shown in Figure 1. $P(V > 0)$ is also known as the Family-wise error rate (FWER).

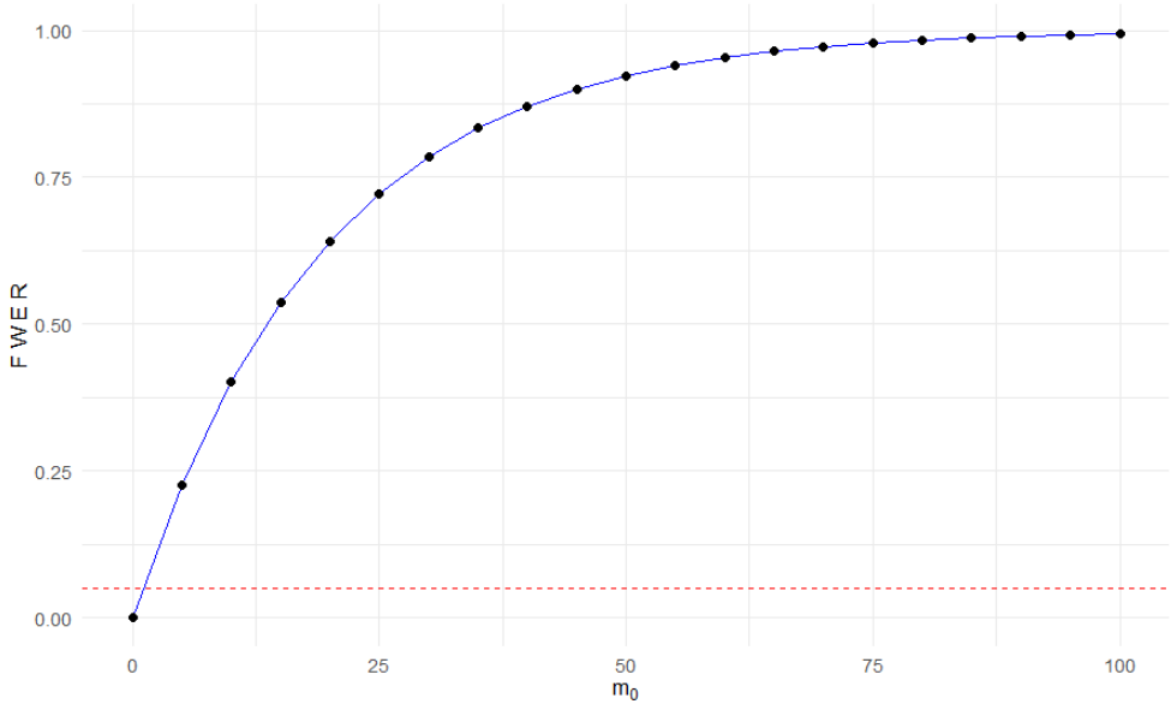

Figure 1: *FWER as function of m_0 under independence. The red line depicts the level of significance $\alpha = 0.05$.*

In order to control the FWER, several procedures exist. A property of multiple testing settings is that in many cases, the covariates correlate with each other. Think for instance of neighbouring pixels in the field of neuroimaging, or genes which are very closely related in the biology setting. In order to keep FWER control stringent without losing much power, the capture of the dependence structure within the covariates is key. Permutation based multiple testing methods takes such dependencies into account. Generally considered the most prominent permutation based multiple testing method for FWER control is the maxT method devised by Westfall and Young (1993). While the maxT method offers good FWER control, there is a downside. The downside to this more complex method is that an additional assumption has to be made. In case of a simple case-control study, this additional assumption is that the joint distribution of the data corresponding to the true hypotheses is the same for cases and controls. In case this assumption is violated (e.g. by

heteroscedasticity), what are the consequences for the FWER control offered by the maxT method? This is a question which remains partly unanswered.

Significance Analysis of Microarrays (SAM) is a different permutation-based multiple testing method which estimates a different error metric than the FWER, namely the False Discovery Proportion (FDP). The FDP is the fraction of false discoveries amongst all rejections. Hemerik and Goeman (2018b) extended SAM by providing a $(1-\alpha)$-confidence upper bound for the estimated FDP. Conveniently, SAM makes the exact same assumption about the data as the maxT method. Hence we may pose a similar unanswered question as we did for maxT: In case this assumption is violated, what are the consequences for the robustness of the $(1-\alpha)$-confidence upper bound estimation?

The main aim of this research is to answer the posed research questions by using simulations in `R`, providing an indication of how robust the methods are.

Before addressing the simulations, some theoretical background knowledge is required. This includes theory on permutation testing; why it is useful, how is it done and which assumptions are made? This will be discussed in Section 2. Building on this in Section 3, theory of multiple testing is discussed. First the multiple testing framework will be discussed, after which the methods and concepts building upon this framework are introduced. This includes the FWER, but also its alternatives such as the False Discovery Rate (FDR) and the False Discovery Proportion (FDP). Multiple testing methodes such as Bonferroni and Holm are also discussed. Afterwards, the more complex permutation-based multiple testing methods maxT and SAM are explained. The main benefits, but also the assumptions and expectation of performance of these methods will be discussed. Section 4 introduces the main novel component to this thesis, namely the setup of the simulations regarding the robustness of the considered methods, and their results.

In Section 5, a data analysis is used to illustrate how the maxT and SAM work in practice on a real data example. The maxT and SAM methods will be applied to a dataset used by Golub et al. (1999) containing gene expressions of two groups of people suffering from different forms of leukemia. Gene expressions could be linked to the difference in leukemia.

Finally, Section 6 is concerned with discussion, and some limitations restricting the study.

2 Permutation Tests

Permutation testing is a non parametric approach to hypothesis testing in which permutations of the observed sample are made in order to construct a permutation distribution of a test statistic assuming the null hypothesis to be true. Based on the permutation distribution and the chosen level of significance, p-values are computed, allowing for the assumed null hypothesis to be either rejected, or not rejected. In the section we go into detail on how permutation testing works using an example, after which we will describe different types of permutation tests. Strictly speaking, the example which will be shown is a randomization test due to the use of randomized treatments (Hemerik & Goeman, 2021). However, this does not matter for the workings of the test itself.

2.1 Permutation example

Suppose a new durable composition of fertilizer has been developed which in theory is designed to increase crop yield. To verify whether or not the fertilizer actually works, a statistical test may be performed. For this test, we assume a controlled environment wherein a single crop grows in a single pot. There are n pots. Some randomly selected pots receive the new fertilizer, whilst others do not. By comparing the crop yield of the two groups of pots, we may assess whether or not the new fertilizer improves crop yield. To formalize this test, we will construct two hypotheses; either the mean crop yield is the same for both groups, or alternatively, the group with the fertilizer has a higher crop yield. These hypotheses are the null hypothesis and the alternative hypothesis respectively, and may be formalised in the following manner:

$$H_0 : \mu_A = \mu_B \qquad H_a : \mu_A > \mu_B \tag{2}$$

Here group A consists of the pots which received the fertilizer, whilst group B consists of the pots which did not. To keep the example straightforward, the amount of pots within the groups is equal, so $\frac{n}{2}$ pots per group. After having received the yield results for each pot, the difference in mean yield produced by pots between the two groups is to be quantified. This quantification comes in the form of a test statistic, in this case the following statistic seems appropriate:

$$T = \sum_{i=1}^{n} \mathbb{1}_A(y_i)x_i - \sum_{i=1}^{n} \mathbb{1}_B(y_i)x_i \tag{3}$$

Here x_i represent the crop yield of pot i, where $i = 1, \ldots, n$. $\mathbb{1}_A(y_i)$ is an indicator function, which takes the value of 1 when pot i is fertilized, otherwise 0. The opposite holds for pots from group B. After defining the test statistic, we may begin to randomly permute group labels to the recorded yields of pots, and recompute the test statistic after

applying a random permutation $\pi \in \prod$, where $\prod$ is the set of all permutations, including those which may be applied to the vector of group labels Y.

Lets say we pick a random subset of permutations $(\pi_1, \ldots, \pi_B) \subset \prod$ and compute the test statistic for each permuted data set. Since we have assumed H_0 to be true, we expect to see no difference between group means under permutation. Therefore, under H_0, it should not matter which label is assigned to which pot. Instead of permuting labels, yields could also instead have been permuted. Hence we need to make an assumption on the data.

Assumption A. $\pi(X) \overset{d}{=} X$, *for all* $\pi \in \prod$.

This somewhat simplified assumption is known as the assumption of exchangeability (Johnson, 1924), which is the prime assumption made in permutation testing. With the B test statistics computed, we may construct the approximate test statistic permutation distribution, which approximates the test statistic values we could have seen under H_0. By observing where our original test statistic falls within this 'null' distribution, we may obtain an indication of the probability of obtaining a similar, or more extreme mean crop yield difference between groups when repeating the experiment, a.k.a. the p-value. The p-value is computed as follows:

$$p = \frac{\#\{k : T^k \geq T^0\}}{B} \tag{4}$$

Here T^0 is the test statistic derived from the unpermuted sample. The p-value is then compared to a predefined level of significance α in order to either reject H_0, or not reject H_0. Because we compare the crop yield means of two independent groups, we assume the variance of the response between the groups to be homogeneous. This independent two sample permutation test is analogous to the independent parametric two samples t-test.

Now the question may arise as to why we would bother following this permutation procedure instead of simply using a independent two samples t-test. The main reason for this is that permuting rows, or labels assigned to rows, maintains the dependence structure within the data, if it is present (Dickhaus, 2014). This is a useful property when data are high-dimensional, and there is some form of dependence between covariates. The importance of this property will become clear in Section 3, when we consider multiple testing settings.

2.2 Types of permutation tests

With a basic example of a permutation test established, we may delve deeper into the types of permutation tests which are available. Berry, Johnston, and Mielke Jr (2014) distinguish three types of permutation tests; the 'complete' permutation test, the resampling-approximation permutation test, and the moment-approximation permutation test.

The first approach is the 'complete' permutation test. It is a permutation test where all of the permutations $\pi \in \prod$, where $|\prod| = n!$, are used to to construct the test statistic permutation distribution. For medium to large sample sizes, the cardinality of $\prod$ becomes too large to evaluate, hence this version of the permutation test is often not feasible to perform. On the other hand, with the advent of high speed computing, the 'complete' test slowly becomes a more realistic option (Berry et al., 2014).

The second approach, a more computationally lenient alternative to the previously mentioned permutation test, is the resampling-approximation to the permutation test. The resampling-approximation randomly selects a few permutations from all the possible permutations $\prod$. This approach is also used in the example in Section 2.1, as $B < n!$. The main benefit is that the permutation distribution under the null is obtained with far less computational demands. In case of a valid exchangeability assumption, and $\prod$ maintaining a group structure as defined by Hoeffding (1952), we may state the permutation distribution to be exact for a large enough, finite B (Hemerik & Goeman, 2018a). Hence this is the most prominently used type of permutation test.

The third approach to permutation testing constitutes of the set of statistical tests which approximate the permutation distribution under the null by using a parametric distribution. One of these tests mentioned by Berry et al. (2014) is the moment-approximation permutation test. This approach requires computation of an arbitrary amount of moments of the test statistic. Using these found moments, a parametric distribution may be selected which approximates the underlying permutation distribution under the null. This approach was mostly used before the widespread use of the resampling-approximation, but still has its merits in case the probability of the observed test statistic is very small. This is because for the resampling-approximation method, the smallest obtainable p-value is limited to $\frac{1}{B}$. P-values of 0 are prevented by always adding the identity permutation to the randomly selected subset of permutations. For permutation tests done in this simulation study, only the resampling-approximation will be used.

When considering settings deviating from the independent samples setting, in particular paired data, adjustments need to be made to make the procedure sketched in Section 2.1 work. To this end we introduce a new experiment. Fisher (1949) analyzed and discussed an experiment designed and conducted by Charles Darwin. Darwin compared the height of two different groups of plants. The first group consisted of plants which fertilized themselves. The second group consisted of plants which required other plants to fertilize them, the so called crossed plants. Each group has n plants. The n plants from each group are paired in this setting due to two plants from differing groups sharing the same pot, watering amount, location etc. Hence some dependence is expected. In order to account for this dependence, the test statistic requires adjusting. The hypotheses in equation (2) still hold in this context if we interpret μ_A as the population mean of the self-fertilizing plant height, and μ_B as the population mean of the crossed plant height. The test statistic is adjusted to this experiment in the following manner:

$$Z = Y_A - Y_B, \quad Z^* = Zg \tag{5}$$

$$T = \frac{\sum_{i=1}^{n} Z_i^*}{n} \tag{6}$$

Z is a vector containing the pairwise differences in height between the groups. The test statistic T now represents the pairwise mean difference between the groups. In order to permute in case of paired data, we can not simply permute the rows of the data, as the dependence between pairs will still remain. The purpose of the permutation is to swap the type of plant the height was measured of, as under H_0, it should not matter if the plant is self-fertilizing or crossed. We can manage this by naturally extending the set of permutations $\prod$ to also include transformations other than solely permutations. The set G also encompasses sign-flipping. A transformation $g \in G$ may be applied similarly to a permutation $\pi \in \prod$. Now we may randomly select a sign-flipping transformation g, which is in essence a vector of length n, where $(g_1, \ldots, g_n) \in \{-1, 1\}^n$. By multiplying the vector containing the random transformation g with the vector of differences Z, we obtain a new vector Z^* over which the test statistic is computed. We repeat this process B times, with B different transformations $g \in G$ to again construct the permutation distribution under the null. To compute T^0, the original test statistic, we leave Z untransformed. The remainder of the process is similar to the example sketched in Section 2.1. This completes the permutation equivalent of the paired t-test.

A minor difference in assumptions compared to the independent setting is present. For the paired setting, we assume a symmetric distribution of the entries of Z (due to sign-flipping transformations) centered around 0 under the null.

A third scenario may present itself in which we have a continuous response and a continuous set of covariate(s). Take for instance the dataset used by Causeur, Friguet, Houee-Bigot, and Kloareg (2011) containing gene expressions of male chickens and their abdominal fatness measured in grams. Some gene expressions are expected to be linked to abdominal fatness. In other words, we wish to verify whether or not there is dependence ($\not\perp\!\!\!\perp$) between certain gene expressions and the abdominal fatness. For illustration purposes we only zoom in on the relation between a single covariate and the response. To formalize this in hypotheses:

$$H_0 : X \perp\!\!\!\perp Y \qquad H_a : X \not\perp\!\!\!\perp Y \tag{7}$$

Here X is a vector containing a single gene expression, and Y a vector of abdominal fatness measurements. In order to quantify the dependence between X and Y, we use the absolute value of the basic correlation coefficient metric as our test statistic. To get our permutation distribution under the null, we again permute. In this setting we permute the response Y for B amount of times using random permutations $\pi \in \prod$. For each permutation π, we recompute the correlation coefficient. With this we again construct the permutation distribution under the null and determine the p-value using equation (4), in which now $T^0 = |\rho(X, Y^0)|$, where Y^0 is the unpermuted measurement of abdominal fatness. With this procedure we may determine if abdominal fatness Y is dependent on gene expression X.

3 Multiple Testing

When testing multiple hypotheses simultaneously, additional problems arise. As described in Section 1, the probability of making type I errors increases drastically with an increasing number of true hypotheses m_0. Simultaneous inference of several covariates, for example genes, without an exploding Type I error rate is desired. In order to reduce the inflation of type I errors, several multiple testing procedures exist. These procedures build upon predefined error metrics relevant for multiple testing. In this section several error metrics, and several procedures controlling or estimating these error metrics will be discussed.

3.1 Error metrics

Suppose we wish to infer m null hypotheses H_j, where $j = 1, \ldots, m$, and $m > 1$. The set of null hypotheses is defined by $(H_1, \ldots, H_m)$. An unknown number of these hypotheses is true (m_0), and an unknown number of these hypotheses is false (m_1). We denote the set of indices belonging to the true hypotheses with $\mathcal{T}$, and the set indices of false hypotheses with $\mathcal{F}$. Clearly $|\mathcal{T}| = m_0$, and $|\mathcal{F}| = m_1$. Within $\{H_j : j \in \mathcal{T}\}$, some hypotheses might be rejected and some not. The number of rejected true hypotheses will be denoted by V. Consequently, the number of not rejected true hypotheses evaluates to $m_0 - V$. For the set $\{H_j : j \in \mathcal{F}\}$, a similar division may be made. The set of indices belonging to rejected hypotheses is denoted by $\mathcal{R} = \{j : p_j \leq t\}$, where t is some unknown cut-off, and $|\mathcal{R}| = R$. The distinction amongst sets of hypotheses, as made by Benjamini and Hochberg (1995), is summarized in Table 2.

Table 2: Contingency table multiple hypothesis testing

	# Not rejected hypothesis	**# Rejected hypothesis**	
# True null hypothesis	$m_0 - V$	V	m_0
# False null hypothesis	$m_1 - U$	U	m_1
	$m - R$	R	m

The variables within the first two rows of the body of Table 2 are unobserved, since m_0 and m_1 are unknown. Only the variables in the bottom row of Table 2 are known. In general, the goal is to find t such that $\mathcal{R}$ matches $\mathcal{F}$ as much as possible. However, similar to rejecting in the single hypothesis setting, errors are bound to be made. An extension of the Type I error to power trade-off sketched in Section 1 occurs when considering multiple inferences. We assume $0 < m_0 < m$ and thus $0 < m_1 < m$. Being conservative when rejecting by lowering t, less hypotheses are rejected, causing the number of Type I errors made (V) to remain low. On the other hand, the cost of being conservative is that the number of Type II errors ($m_1 - U$) increases. Hence the off-diagonal in Table 2 displays the number of Type I and Type II errors respectively. Thus determining $\mathcal{R}$ is essential. Finding a set $\mathcal{R}$ whilst keeping V low and maintaining U may be done by a rejection procedure

which constraints itself based on an error metric. The first error metric we discuss is the Family Wise Error Rate (FWER), which we have seen before in Section 1. The FWER is defined as the probability of making at least one false inference.

$$FWER = P(V > 0) \tag{8}$$

Under independence of hypotheses, the FWER among true hypotheses assumes the form as shown in equation (1). The second error metric is the False Discovery Rate (FDR).

$$FDR = \begin{cases} E\left(\frac{V}{R}\right) & \text{when } R \geq 1 \\ 1 & \text{when } R = 0 \end{cases} \tag{9}$$

The FDR is the expectation of the False Discovery Proportion (FDP). It is a metric which expresses the expected proportion of type I errors among all of the rejected hypotheses. Both error metrics are generalizations of the Type I error. They are however not the same.

The first, and most important distinction is that FWER protects against the risk of incurring any Type I error at all, whilst FDR allows for some Type I errors, as long as on average most findings are valid. This results in FWER being more conservative at the cost of power relative to FDR, which is somewhat less conservative whilst having more power. In the case of $\mathcal{F} = \emptyset$, it follows that $FDR = FWER = 0$, because $m = m_0$.

The second distinction considers subsets of hypotheses. For any subset of rejected hypotheses, FDR control is no longer valid. Interpretation of FDR is hence restricted to the context of the total set of features tested. The lack of the subsetting property for FDR control is a product of the averaging inherent to FDR. The FWER does not suffer from this drawback.

The third distinction considers practical uses. Take for instance an experiment where one wishes to find deferentially expressed genes in relation to some phenotype using an FDR controlling procedure. Benjamini, Drai, Elmer, Kafkafi, and Golani (2001) state that FDR control in this setting is mainly useful for screening purposes, meaning that the hypotheses rejected by the FDR controlling procedure are likely to represent deferentially expressed genes, although with more uncertainty, due to the averaging inherent to the error metric. FWER is preferred when the number of tests is relatively small and no further validation of hypotheses is available.

3.2 FWER controlling procedures

There exist many procedures to control and/or estimate the FWER, FDP and FDR. Several are presented in here. They differ in their assumptions and computation steps. More restrictive assumptions generally yield more powerful procedures but limit their applicability to certain data configurations. The FWER controlling procedures which will be discussed include Bonferroni, Holm and the maxT method. Comparisons between the procedures regarding their FWER controlling capability will discussed in Section 4. FDP estimation will only be covered by the Significance Analysis of Microarrays (SAM) estimation method and its $(1-\alpha)$-confidence upper bound extension. FDR controlling methods will not be discussed as they fall outside the scope of the research questions posed in Section 1.

The first, and also the most intuitive FWER controlling procedure is the Bonferroni correction. The Bonferroni correction does not require any assumptions, but does suffer from being too conservative in a number of settings. The procedure is displayed in algorithm block 1.

Algorithm 1 Bonferroni rejection procedure

```
for j in 1, ..., m do
    if p_j ≤ α/m then
        Reject H_j
    else if p_j > α/m then
        Do not Reject H_j
    end if
end for
```

The rejection threshold t is defined as α/m. Because Bonferroni commits a type I error only if $p_j \leq \alpha/m$, for $j \in \mathcal{T}$, FWER control is dependent on the proportion of true hypotheses amongst all hypotheses. Hence the FWER is not controlled at the level α, but rather at $m_0\frac{\alpha}{m}$, which is a stricter level than α. Therefore in case of a setting where $m_1 >> m_0$ and one opts to control the FWER using bonferroni, FWER control will be strict, causing only few rejections. This also happens amongst the false hypotheses, resulting in more type II errors. Bonferroni is also conservative when many dependencies amongst covariates are present (Abdi et al., 2007).

The second FWER controlling procedure is the Holm correction, wherein Holm attempted to remedy the frequently occurring limited power inherent to Bonferroni (Holm, 1979). This comes at the price of a slightly more complex procedure. Similar to Bonferroni, Holm does not require any prior assumptions to be made, and may be applied under any dependence structure. Contrary to Bonferroni, Holm often does control FWER at a level closer to α, causing the procedure to be less conservative than Bonferroni. Hence Holm is almost always preferred over Bonferroni. The procedure is shown in algorithm block 2.

Algorithm 2 Holm rejection procedure

1 Sort all p-values $p_1 \dots p_m$ in ascending order, where for p_j, j indicates rank
2 $p_1^{adj} = p_1 \cdot m$
3 **for** j in $2, \dots, m$ **do**
4 $p_j^{adj} = max(p_{j-1}^{adj}, p_j \cdot (m + 1 - j))$
5 **if** $p_j^{adj} \leq \alpha$ **then**
6 Reject H_j
7 **else if** $p_j^{adj} > \alpha$ **then**
8 Do not Reject H_j
9 **end if**
10 **end for**

Instead of adjusting the rejection threshold, this time the p-values themselves are adjusted. The procedure could also have been done by adjusting the rejection threshold sequentially, but it does not make a difference for determining $\mathcal{R}$. The same may also be done for Bonferroni.

Line 4 enforces monotonicity of the sequence of the adjusted p-values, such that p-values will never change in rank post adjusting. Since each p_j^{adj} is dependent on its rank j, which in turn is dependent on all $p_{k<j}$, monotonicity amongst adjusted p-values is desired. It would be detrimental to have a hypothesis higher in rank than H_j possibly rejected, whilst H_j remains unrejected.

Similar to Bonferroni, Holm still suffers from being conservative when many positive correlations are present. As mentioned before, this is undesired in neuroimaging data (neighboring pixels may be extremely highly correlated), or DNA genomics data where gene activity may be very similar regarding a specific phenotype. The next FWER controlling procedure takes this dependence into account through the use of permutations.

The third FWER controlling procedure is the maxT procedure devised by Westfall and Young (1993). The maxT procedure differs greatly from the previous two correction procedures, as it is strictly permutation based. In essence the $(1-\alpha)$-quantile of the distribution of the maximum test statistic value of the m_0 true hypotheses is found through permuting the observed data. This quantile is then used as rejection threshold t.

The use of permutations has its benefits, but also requires stricter assumptions. We consider data of the following form, where the presence of the vector Y is optional.

$$X_{n,m} = \begin{pmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,m} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n,1} & x_{n,2} & \cdots & x_{n,m} \end{pmatrix}, \; Y_n = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}$$

Assumption B. *We assume the data matrix is of the form $X = (X_1, \ldots, X_m)$, where we assume for every $1 \leq j \leq m$, the test statistic $T_j(X_j)$ depends only on X_j. Then for any nonempty set $M \subseteq \{1, \ldots, m\}$, denote $X_M = (X_j : j \in M)$. We assume $X_{\mathcal{T}} \overset{d}{=} \pi X_{\mathcal{T}}$ for every $\pi \in \prod$, where $\mathcal{T} = \{j \in M : H_j = true\}$ and assumed nonempty. In case of presence of the data vector Y, we assume that conditional on Y, $X_{\mathcal{T}} \overset{d}{=} \pi X_{\mathcal{T}}$ for every $\pi \in \prod$, where $T_j(X_j, Y)$ depends only on X_j and Y.*

Recall the permutation example described in Section 2.1. In the two sample permutation example, group labels were assumed to be permutation invariant under H_0 through the exchangeability assumption. We extend this notion here. In essence we assume the data belonging to the true hypotheses ($X_{\mathcal{T}}$) to be permutation invariant. This general assumption may take different forms, depending on whether or not a vector Y is present containing group labels, or a continuous outcome. Hence a small extension was required.

$X_{\mathcal{T}} \overset{d}{=} \pi X_{\mathcal{T}}$ indicates π is a null-invariant transformation of $X_{\mathcal{T}}$, meaning the joint distribution of $X_{\mathcal{T}}$ does not change under permutation. In a case-control setting we would therefore have to assume that the joint distribution of the part of the data corresponding to the true hypotheses for the cases is the same as for the controls and vice-versa, otherwise the permutation π would in theory not be null-invariant. This holds for permutation based multiple testing in general, and is not limited to any controlling method. However, assuming identical distributions might not always be realistic. Think for instance of a different dependence structure between the cases and controls, or heteroscedasticity (Goeman & Solari, 2014). In Section 4 we zoom in on the robustness of FWER control offered by the maxT procedure in case of violations of this assumption. The maxT rejection procedure is shown in algorithm block 3.

Algorithm 3 maxT rejection procedure

$T_1^0, \ldots, T_m^0$ are the test statistics $T_j(X_j, Y)$ of the original observed sample.
for b in $1, \ldots, B$ **do**
    $X = \pi_b(X_0)$
    for j in $1, \ldots, m$ **do**
        $\mathbf{T}_j^b = T_j(X_j, Y)$, where $\mathbf{T}$ is a $B \times m$ matrix.
    end for
end for

Initialize while loop
while rejections are being made **do**
    for b in $1, \ldots, B$ **do**
        $T_{max} = max(\mathbf{T}_*^b)$, where T_{max} is a vector of the largest test statistics.
    end for
    sort T_{max} in ascending order
    define q as the $(1 - \alpha)$ quantile corresponding to T_{max}
    for j in $1, \ldots, m$ **do**
        if $T_j^0 > q$ **then**
            Reject H_j
        else if $T_j^0 \leq q$ **then**
            Do not Reject H_j
        end if
    end for
    Update $\mathcal{R}$
    Remove columns from $\mathbf{T}$ corresponding to $\mathcal{R}$
end while

The algorithm iteratively constructs a different distribution of the maximum test statistics for the hypotheses still in play. In each iteration the test statistics of the original sample are compared to the $(1 - \alpha)$-quantile of maximum test statistics distribution. Hypotheses with test statistics exceeding the $(1 - \alpha)$-quantile are rejected. Afterwards $\mathbf{T}$ is updated according to $\mathcal{R}$ to remove permutation test statistics corresponding to the rejected hypotheses. This in turn makes the new distribution of the maximum test statistics less wide, resulting in a lower $(1-\alpha)$-quantile, allowing for new possible rejections. This causes the rejection procedure to narrow down, and eventually subside when there are no more outlying original test statistic relative to the distribution of the maximum test statistics.

There are several versions of the maxT procedure. A widely used version of maxT is the single-step procedure, which is often used since it is computationally more feasible, allows for easy adjusted p-values and in most settings the FWER control is comparable to the FWER control provided by the iterative rejection procedure. The complete procedure does in theory remain more powerful. The used single-step procedure is discussed more in

depth by Ge, Dudoit, and Speed (2003).

3.3 FDP estimation through SAM

Significance Analysis of Microarrays (SAM) is a permutation-based multiple testing method devised by Tusher, Tibshirani, and Chu (2001) which provides and estimate for the False Discovery Proportion (FDP). FDP is defined by the fraction of false discoveries amongst all rejections. Contrary to Westfall & Young' maxT method, SAM does not offer control over any Type I error metric, but simply provides an estimate of the prevalence of Type I errors expressed by the metric. Tusher et al. (2001) have shown that SAM may provide different estimates for the FDP compared to conventional methods of analysis which estimate the FDP. Hence it is a method worth considering in the context of permutation-based multiple testing.

SAM provides the FDP estimate as follows. SAM rejects all hypotheses with test statistics or p-values lying within a user-defined rejection region D. The number of false positives amongst all inferences (V) is estimated through permutation of the data. For each version of the data where permutation π is applied to the data X, rejections are made based on the user-defined rejection region. Taking the median of the numbers of rejections made for all $(\pi_1, \ldots, \pi_B)$ applied to X provides an estimate for V. Dividing this median by the number of rejections is then an estimate of the FDP.

Hemerik and Goeman (2018b) extended SAM by providing a $(1-\alpha)$-confidence upper bound for the estimated FDP. The upper bound for the estimated number of false discoveries is defined as follows. We predefine $a \wedge b$ as the minimum between a and b. $\overline{V} := R^k \wedge R$, where $\overline{V}$ is the upper bound for V. R^k is the $(1-\alpha)$-quantile of the numbers of rejections for the permuted versions of the data. The $(1-\alpha)$-confidence upper bound for the estimated FDP, defined as $\overline{FDP}$, is then found by dividing $\overline{V}$ by R. The use of $R^k \wedge R$ as an upper bound for V seems intuitive as it always holds that $V \leq R$. A more formal proof on why this definition of $\overline{FDP}$ is indeed a valid $(1-\alpha)$-confidence upper bound for the estimated FDP may be found in Hemerik and Goeman (2018b).

4 Simulations

With simulations we attempt to answer the two questions posed in the introduction. To recap, the questions were:

1. In case assumption B is violated, what are the consequences for the FWER control offered by the maxT method?

2. In case assumption B is violated, what are the consequences for the validity of the $(1-\alpha)$-confidence upper bound estimation using SAM?

Assumption B is extensively described in Section 3.2. We narrow this general assumption down by only considering case-control data. Two separate settings were mentioned in which the assumption would be violated in case of case-control data for the simulations. The first setting which violates assumption B has to do with a different dependence structure between covariates between the cases and the controls. The second setting considers heteroscedasticity in the data between case and control covariates. Here we add a third setting, in which the joint distribution from which the cases are generated is a completely different distribution from which the controls are generated (e.g. normal vs exponential).

4.1 Base settings

As mentioned, we separate the possible ways the violate assumption B for both methods in several simulation settings. The simulation settings may be found in Table 3. Simulation setting 3 attempts to remedy probable issues caused by heteroscedasticity by using a different test statistic. Each simulation setting will be laid out individually.

Table 3: Simulation settings

No.	**Method**	**description of simulated data**
1	maxT	Different joint distributions due to varying dependence structure
2	maxT	Different joint distributions due to heteroscedasticity
3	maxT	Use of Welch t-statistic in case of heteroscedasticity
4	maxT	Different joint distributions due to different marginal distributions
5	SAM	Different joint distributions due to varying dependence structure
6	SAM	Different joint distributions due to heteroscedasticity

The data generating process is mostly the same in all settings, with some slight tweaks. Figure 2 provides a visual representation on how the data are structured.

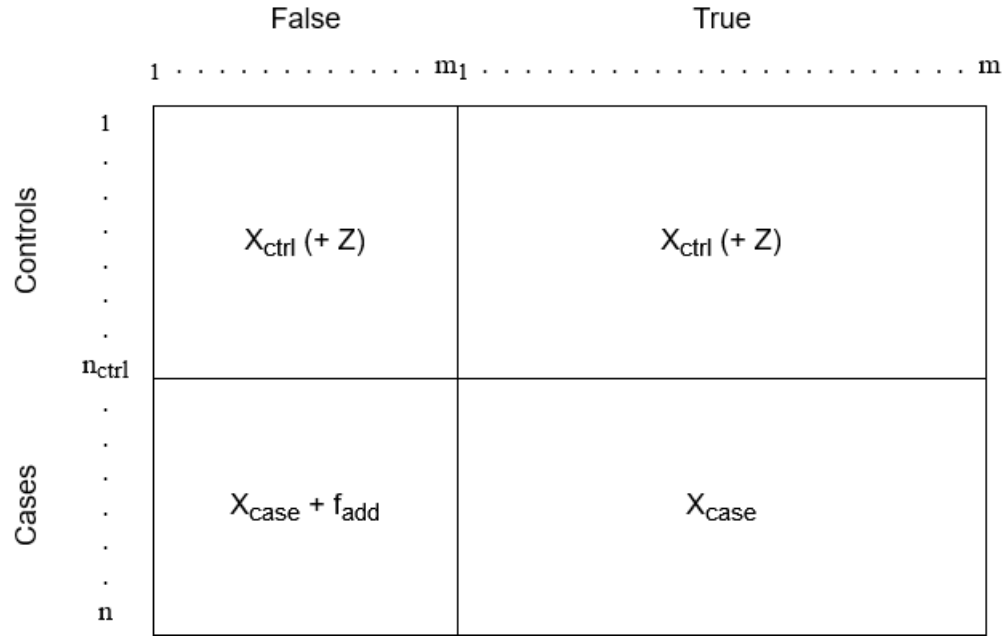

Figure 2: *A simplified representation of the generated data. The data consists of two vertically stacked matrices containing control data X_{ctrl}, and case data X_{case}. Z is a vector added to control data to simulate dependence between covariates. f_{add} is a value used to make hypotheses false by adding it to each covariate belonging to the set of false hypotheses and case observations.*

For all simulations settings, we keep the number of observations $n = 20$ constant, where $n_{ctrl} = n_{case} = \frac{n}{2}$. The number of hypotheses m is also being kept constant, where the number of false hypotheses $m_1 = f_{prop} \cdot m$ and the number of true hypotheses $m_0 = m - m_1$. The proportion of false hypotheses is also set for all simulations at $f_{prop} = 0.1$. Therefore $m_1 = 50$, and $m_0 = 450$ are also constants across the different simulations. Assuming the context of microarray data analysis for the simulations, for each gene j, the null hypothesis H_j is defined as the hypothesis that gene j is not differentially expressed.

4.2 Simulation setup and results

In this subsection the simulation configurations are specified, after which the simulation results are shown and briefly discussed. Each of the simulations correspond to their respective `R`-scripts found in Appendix A.

4.2.1 maxT: Heterogeneous dependence

In the first simulation setting we attempted to examine how robust the maxT method is when the cases and controls have non-identical dependence structures. Going back to Figure 2 we specify the following data generating distributions for this simulation:

$\mathbf{X_{ctrl}} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, where $\boldsymbol{\mu}$ is a vector of length m filled with zeros, and $\boldsymbol{\Sigma}$ a $m \times m$ diagonal variance-covariance matrix, with on the diagonal $\sigma^2_{ctrl} = 1$. $\mathbf{X_{case}} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, where $\boldsymbol{\mu}$ is a vector of length m filled with zeros, and $\boldsymbol{\Sigma}$ a $m \times m$ diagonal variance-covariance matrix, with on the diagonal $\sigma^2_{case} + \sigma^2_Z$, where $\sigma^2_{case} = 1$. We sampled n_{ctrl} and n_{case} samples from their respective multivariate normal distributions to generate the required data.

In order to mimic dependence, we row-wise added a random vector $\mathbf{Z}$ of size n_{ctrl} to the controls, where $\mathbf{Z} \sim \mathcal{N}(\mu = 0, \sigma^2_Z)$. This means that the value z_1 was added to all covariates of the first observation, z_2 to all covariates of the second observation etc. The strength of the dependence between the covariates of the control group was measured using the correlation coefficient:

$$\rho_{ctrl} = \frac{\sigma^2_Z}{\sigma^2_Z + \sigma^2_{ctrl}} \tag{10}$$

There was no dependence simulated between the covariates of the cases. A minor oddity is that for the cases the variance-covariance matrix does not have σ^2_{case} on the diagonal, but rather $\sigma^2_{case} + \sigma^2_Z$. This was done to keep the variances of the cases and controls identical. In this way we may discern the dependence setting from the heteroscedasticity setting without confounding them.

Next we needed to make the false hypotheses actually false. This was done by adding a value f_{add} to the data corresponding to the first m_0 hypotheses marked as false, but only for the cases. f_{add} is defined as a product comprising of two parts. The first factor is a simple effect size, denoted by ϕ, which is set as a constant throughout all simulations at $\phi = \sqrt{2}$. The second factor is a scalar. This scalar is defined as $\sqrt{\sigma^2_{ctrl} + \sigma^2_{case} + 2\sigma^2_Z}$. The purpose of this scalar is to keep the signal proportional to the noise. Hence:

$$f_{add} = \phi\sqrt{\sigma^2_{ctrl} + \sigma^2_{case} + 2\sigma^2_Z} \tag{11}$$

To keep signal-to-noise ratio constant, we took the effect size proportional to the standard deviations of the observations. If f_{add} would not scale appropriately with σ^2_Z, the

signal would be drowned out for larger simulated values of σ_Z^2, resulting in a confounded depiction of statistical power. With a constant signal-to-noise ratio, we may observe the effects of different dependence structures on power in a more objective manner.

With the simulated data in place, we applied the single-step maxT, sequential maxT and Holm procedures to the data. For the single-step maxT procedure the `R` package *multtest* by Pollard, Dudoit, and van der Laan (2005) was used. For this procedure and Holm, the number of permutations was set to $B = 20000$. For the sequential maxT procedure based on algorithm 3, only $B = 20$ permutations were used due to computational constraints. The test statistic which was used is the independent two samples t-test statistic for two-sided tests:

$$T = \left| \frac{\overline{X_{ctrl}} - \overline{X_{case}}}{S_p \sqrt{\frac{1}{n_{ctrl}} + \frac{1}{n_{case}}}} \right| \text{ where } S_p = \sqrt{\frac{(n_{ctrl} - 1)S^2_{X_{ctrl}} + (n_{case} - 1)S^2_{X_{case}}}{n_{crtl} + n_{case} - 2}} \tag{12}$$

The degrees of freedom of the tests was set to $df = n_{ctrl} + n_{case} - 2$, and the significance level to $\alpha = 0.05$. Eventually, the single-step maxT and Holm adjust p-values were estimated by their respective procedures, after which we compared each adjusted p-value to the threshold α. For the sequential method, rejections were made without p-values as depicted in algorithm block 3. Since we know which hypotheses belonged to the set of false hypotheses $\{H_j : j \in \mathcal{F}\}$, we computed how many false positives were produced by the procedures. The number of Type II errors was computed in a similar fashion.

Starting from generating the data to computing the number of Type I and Type II errors computed by each procedure was considered a single iteration. We repeated this sequence of steps $r = 5000$ times, for six different values of $\sigma_Z^2 = (0, 0.4, 0.8, 1.2, 1.6, 2)$. The correlations $\rho_{ctrl} = (0.00, 0.29, 0.44, 0.55, 0.62, 0.67)$ between the hypotheses of the controls correspond to the different variance values σ_Z^2 took.

In order to obtain an estimate for FWER control offered by maxT and Holm in these different dependence settings, we used the following:

$$\widehat{FWER} = \frac{\#\{k : V_k > 0\}}{r} \tag{13}$$

We denote V_k as the number of false positives in iteration k. Since $\#\{k : V_k > 0\} \sim Bin(\widehat{FWER}, r)$, and $r = 5000$, we used the normal-approximation to the binomial in order to produce a $(1 - \alpha)$-confidence interval for the FWER estimates.

The type II error rate ($\widehat{\beta}$) was estimated by dividing the count of all Type II errors committed across all r iterations by the count of all hypotheses within those r iterations for which a Type II error could have been committed. The estimated, and also reported power is defined by $1 - \widehat{\beta}$. The results are shown in Table 4.

Table 4: Estimates as function of different dependence

ρ_{ctrl}	$\widehat{FWER}_{single}$	$1-\widehat{\beta}_{single}$	$\widehat{FWER}_{seq}$	$\widehat{FWER}_{Holm}$	$1-\widehat{\beta}_{Holm}$
0.000	0.047 ± 0.006	0.380	0.047 ± 0.006	0.049 ± 0.006	0.314
0.286	0.042 ± 0.006	0.386	0.046 ± 0.006	0.046 ± 0.006	0.315
0.444	0.044 ± 0.006	0.392	0.047 ± 0.006	0.043 ± 0.006	0.316
0.545	0.044 ± 0.006	0.401	0.049 ± 0.006	0.044 ± 0.006	0.315
0.615	0.044 ± 0.006	0.408	$\mathbf{0.054} \pm 0.006$	0.042 ± 0.006	0.315
0.667	0.043 ± 0.006	0.416	$\mathbf{0.057} \pm 0.006$	0.040 ± 0.005	0.315

The Bonferroni correction was omitted due to Holm being uniformly more powerful than Bonferroni. Eestimates for which $\widehat{FWER} > \alpha$ are **boldfaced**.

The Holm correction was done on raw p-values. Raw p-values are simply permutation p-values obtained from the procedure in Section 2.1 applied to all covariates j, using a two-sided test and the test statistic specified in equation (12). Going back to algorithm block 2, line 2, we can see that the Holm correction has its smallest adjusted p-value set to $p_1^{adj} = p_1 \cdot m$. Since the identity permutation is always included in the set of permutations, the smallest possible raw p-value is $\frac{1}{B}$, hence the smallest possible Holm adjusted p-value is $\frac{m}{B}$. Therefore it was of importance to not take B too low, as Holm would break down in our current simulation setting if $B < 10000$.

What becomes apparent from Table 4 is that FWER-control offered by the single-step maxT procedure does not suffer from cases and control having a different dependence structure, as $\widehat{FWER} < \alpha$ holds under increasingly correlated covariates for controls. The same could be said for FWER-control offered by Holm, but this was expected as Holm does not make any assumptions on the dependence structure of the data. Statistical power also seems relatively unfazed. When comparing the performance of the two methods we can see relatively similar results, however the single-step maxT method does appear to be more powerful than Holm, as was expected. Remarkable is the discrepancy in power between maxT and Holm when considering no dependence, as we would expect the methods to behave similar in this situation. This discrepancy in power could be attributed to the limited amount of permutations relative to the number of hypotheses.

The large difference in the number of permutations forms a problem when comparing the sequential maxT' power performance to the other methods. Hence we leave this comparison out. Even though $\widehat{FWER} > \alpha$ strictly no longer holds for stronger correlations for the sequential method, the $FWER$ is still close to α. Contrary to single-step maxT, the sequential maxT $FWER$ increases with an increasing correlation. This could be due to the iterative nature of the method, causing more false positives in a second or third rejection iteration.

We also simulated increasing correlation under sample size imbalance and found sample size imbalance to not cause $\widehat{FWER} > \alpha$ to be violated for Holm and the single-step approach. For the sequential method it did, similar to the results shown in Table 4.

4.2.2 maxT: Heteroscedasticity

In the second simulation setting we attempted to examine how robust the maxT method is under heteroscedasticity. Several simulation settings differ from the previous simulation. For the cases, $\mathbf{\Sigma}$ now has $\sigma^2_{case} = 1$ on the diagonal. For the controls, σ^2_{ctrl} is no longer constant. The variance of the controls takes the values $\sigma^2_{ctrl} = (1, 1.8, 2.6, 3.4, 4.2, 5.0)$. Moreover, we omitted any form of dependence to keep the two settings separate as much as possible. Hence the dependence-inducing vector $\mathbf{Z}$ was no longer required. This means that the scalar factor of f_{add} had to be adjusted into $\sqrt{\sigma^2_{ctrl} + \sigma^2_{case}}$. The effects of heteroscedasticity are displayed in table 5.

Table 5: Estimates as function of increasing heteroscedasticity

σ^2_{ctrl}	$\widehat{FWER}_{single}$	$1 - \widehat{\beta}_{single}$	$\widehat{FWER}_{seq}$	$\widehat{FWER}_{Holm}$	$1 - \widehat{\beta}_{Holm}$
1.0	0.047 ± 0.006	0.380	0.047 ± 0.006	0.049 ± 0.006	0.314
1.8	$\mathbf{0.064} \pm 0.007$	0.398	$\mathbf{0.060} \pm 0.007$	$\mathbf{0.058} \pm 0.006$	0.325
2.6	$\mathbf{0.086} \pm 0.008$	0.427	$\mathbf{0.085} \pm 0.008$	$\mathbf{0.072} \pm 0.007$	0.341
3.4	$\mathbf{0.119} \pm 0.009$	0.453	$\mathbf{0.110} \pm 0.009$	$\mathbf{0.086} \pm 0.008$	0.356
4.2	$\mathbf{0.154} \pm 0.010$	0.476	$\mathbf{0.139} \pm 0.010$	$\mathbf{0.104} \pm 0.008$	0.369
5.0	$\mathbf{0.187} \pm 0.011$	0.495	$\mathbf{0.173} \pm 0.010$	$\mathbf{0.121} \pm 0.009$	0.380

Clearly under increasing heteroscedasticity $\widehat{FWER} < \alpha$ is no longer valid, and all methods become very anti-conservative. As expected, due to violation of assumption B, FWER control provided by maxT is no longer reliable. Holm also breaks down. This is the case due to the use of permutations assuming exchangeability, which is not longer valid due to heteroscedasticity. Hence even though the Holm procedure is assumption-free, the procedure by which the permutation p-values were obtained is not. If p-values we computed using a parametric test allowing for heteroscedasticity, the Holm procedure would likely not produce results as anti-conservative as displayed in Table 5.

Even though the FWER is not the same as the Type I error rate, the Trade-off between Type I and Type II errors sketched in Section 1 is clearly displayed in Table 5. Since the FWER increases due to increasing heteroscedasticity, the power of the test increases as well.

4.2.3 maxT: Heteroscedasticity (Welch)

Heteroscedasticity has proven to be detrimental for FWER control. A possible solution worth exploring was the Welch's t-test statistic devised by Welch (1947) in order to have the Student's t-test adapt to unequal variances and/or unequal sample sizes. It also serves as an approximate solution to the famous Behrens-Fisher problem.

For this third simulation setting we kept most settings similar to the previous simulation. We did however swap the conventional two-sample t-test statistic for Welch's t-test statistic which is defined as follows:

$$T = \left| \frac{\overline{X_{ctrl}} - \overline{X_{case}}}{S_{\bar{\Delta}}} \right| \text{ where } S_{\bar{\Delta}} = \sqrt{\frac{S^2_{X_{ctrl}}}{n_{ctrl}} + \frac{S^2_{X_{case}}}{n_{case}}} \tag{14}$$

The degrees of freedom also varies from the conventional two-sample t-test statistic. Most notable is that the Welch's t-test statistic does not depend on a pooled variance estimate, allowing for different variances between populations. Since the Welch's t-test statistic and the conventional two-sample t-test produce identical test statistics under equal sample sizes, we introduced some skew in sample size between the cases and controls to properly gauge the effectiveness of the Welch's t-test statistic . We set $n_{ctrl} = 8$ and $n_{case} = 12$, instead of having equal sample sizes.

Table 6: Single-step maxT estimates as function of increasing heteroscedasticity and different test statistics

σ^2_{ctrl}	$\widehat{FWER}_T$	$1 - \widehat{\beta}_T$	$\widehat{FWER}_{Welch}$	$1 - \widehat{\beta}_{Welch}$
1.0	0.044 ± 0.006	0.356	0.041 ± 0.005	0.330
1.8	$\mathbf{0.101} \pm 0.008$	0.420	$\mathbf{0.073} \pm 0.007$	0.329
2.6	$\mathbf{0.193} \pm 0.011$	0.474	$\mathbf{0.130} \pm 0.009$	0.345
3.4	$\mathbf{0.306} \pm 0.013$	0.515	$\mathbf{0.195} \pm 0.011$	0.364
4.2	$\mathbf{0.407} \pm 0.014$	0.546	$\mathbf{0.257} \pm 0.012$	0.381
5.0	$\mathbf{0.504} \pm 0.014$	0.572	$\mathbf{0.322} \pm 0.013$	0.397

From Table 6 it becomes apparent that the Welch's t-test statistic did not solve the problems heteroscedasticity causes for the FWER control of the single-step maxT method. maxT still becomes very anti-conservative. However, when comparing the FWER estimates using the conventional t-test statistic and Welch's test statistic, the Welch test statistic did seem to be a bit less liberal than its counterpart. When we compared FWER estimates of Table 6 to those of Table 5, it became clear that the difference in sample size between the cases and controls had caused a substantial increase in the estimated FWER. This was expected as according to Lehmann, Romano, and Casella (2005), the two-sample t-test performs relatively well under heteroscedasticity, as long as the samplesizes are balanced.

Table 7: Sequential maxT estimates as function of increasing heteroscedasticity and different test statistics

σ^2_{ctrl}	$\widehat{FWER}_T$	$\widehat{FWER}_{Welch}$
1.0	0.045 ± 0.006	0.046 ± 0.006
1.8	$\mathbf{0.103} \pm 0.008$	$\mathbf{0.079} \pm 0.007$
2.6	$\mathbf{0.202} \pm 0.011$	$\mathbf{0.131} \pm 0.009$
3.4	$\mathbf{0.328} \pm 0.013$	$\mathbf{0.201} \pm 0.011$
4.2	$\mathbf{0.484} \pm 0.014$	$\mathbf{0.273} \pm 0.012$
5.0	$\mathbf{0.655} \pm 0.013$	$\mathbf{0.351} \pm 0.013$

Table 7 tells a story for the sequential maxT procedure akin to Table 6 for the single-step maxT procedure. We again see Welch's t-test statistic did not solve the problems heteroscedasticity causes for FWER control, but did somewhat improve on using the two sample t-test statistic.

Table 8: Holm estimates as function of increasing heteroscedasticity and different test statistics

σ^2_{ctrl}	$\widehat{FWER}_T$	$1 - \widehat{\beta}_T$	$\widehat{FWER}_{Welch}$	$1 - \widehat{\beta}_{Welch}$
1.0	$\mathbf{0.053} \pm 0.006$	0.314	0.048 ± 0.006	0.291
1.8	$\mathbf{0.115} \pm 0.009$	0.383	$\mathbf{0.082} \pm 0.008$	0.313
2.6	$\mathbf{0.191} \pm 0.011$	0.426	$\mathbf{0.122} \pm 0.009$	0.338
3.4	$\mathbf{0.265} \pm 0.012$	0.453	$\mathbf{0.162} \pm 0.010$	0.361
4.2	$\mathbf{0.342} \pm 0.013$	0.474	$\mathbf{0.202} \pm 0.011$	0.381
5.0	$\mathbf{0.409} \pm 0.014$	0.489	$\mathbf{0.245} \pm 0.012$	0.398

Table 8 shows Holm did not maintain FWER control either, even though Welch's test statistic was used. This was unexpected as Janssen (1997) has shown permutation tests based on studentized statistics to be asymptotically exact at the level $\alpha = 0.05$. However, Janssen (1997) did not take the multiple testing setting into account, wherein Holm rejects hypotheses at a far more conservative level than α. Though both methods failed here, it is noteworthy that Holm has outperformed maxT in terms of FWER in almost all heteroscedasticity settings.

4.2.4 maxT: Different distributions

In the fourth simulation setting we attempted to gauge the FWER control offered by maxT when the joint distributions of the cases and controls vary completely. No dependence, heteroscedasticity or sample size differences were simulated. There are however differences between the cases and controls. $\mathbf{X_{ctrl}} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, where $\boldsymbol{\mu}$ is a vector of length m filled with zeros, and $\boldsymbol{\Sigma}$ a $m \times m$ diagonal variance-covariance matrix, with on the diagonal $\sigma^2_{ctrl} = 1$. We drew n_{ctrl} random samples from the multivariate normal to generate control data. $\mathbf{X_{case}} \sim Exp(\lambda)$, where $\lambda = 1$. We drew $m \cdot n_{ctrl}$ random samples from the defined exponential distribution to generate the data for the cases. In order to maintain $E(\mathbf{X_{ctrl}}) = E(\mathbf{X_{case}})$, we zero centered the cases by subtracting $E(\mathbf{X_{case}})$ from each entry of $\mathbf{X_{case}}$. The scalar factor of f_{add} was also altered into $\sqrt{\sigma^2_{ctrl} + \frac{1}{\lambda^2}}$ to keep the signal-to-noise ratio constant.

Since we did not vary any correlation or variance, we only have singular estimates. $\widehat{FWER}_{single} = \mathbf{0.237} \pm 0.014$. From this we may conclude that the single-step maxT becomes anti-conservative, similar to the heteroscedasticity setting. $1 - \widehat{\beta}_{single} = 0.453$ also reflects this. For the sequential method $\widehat{FWER}_{seq} = \mathbf{0.231} \pm 0.012$. For Holm $\widehat{FWER}_{Holm} = \mathbf{0.165} \pm 0.010$. Again Holm also becomes anti-conservative, and yet again less so than maxT. Unexpected was that the power $1 - \widehat{\beta}_{Holm} = 0.498$ was higher than that of the maxT method. Hence in this specific setting Holm seems superior in terms of a lower FWER as well as a higher power. This is the case due to the lack of correlation in the data.

4.2.5 SAM: Heterogeneous dependence

In the fifth simulation setting we attempted to examine the robustness of the $(1-\alpha)$-confidence upper bound estimation using SAM under different dependence structures. We express this robustness through $P(\overline{FDP} < FDP)$.

The generated data was exactly the same as in the first simulation setting. Similar to step 5 in algorithm block 3, a matrix of permutation test statistics using (12) as the test statistic was constructed. Since the test statistics follow a T distribution, they were easily converted into p-values. We call this matrix of p-values $\mathbf{P}$. This was done so $\mathbf{P}$ could be used as input for the SAM procedure. The SAM procedure and the $(1-\alpha)$-confidence upper bound estimation was done using the `R` package *confSAM* by Hemerik and Goeman (2018b). For each hypothesis the rejection region D was of the the form $(0, c)$, where $c = 0.01$ was set to be constant. $\alpha = 0.05$ was set such that $(1-\alpha)$ was the desired confidence level. Therefore we require $P(\overline{FDP} < FDP) \leq \alpha$.

Since Holm was no longer included in the simulations, we were able to use fewer permutations to speed up computation. We set the number of permutations to $B = 100$. For a larger amount of permutations results are very similar (Marriott, 1979). Similar to the previous simulations, each iteration of a single simulation includes generating the data and estimating the desired metrics, which in this case are $\overline{FDP}$ and FDP. $\overline{FDP}$ was estimated using *confSAM*, whereas the FDP was found using the first row of p-values from $\mathbf{P}$. This first row contained the p-values of all j hypotheses of the identity permutation. Since we knew which hypotheses were actually false, and which hypotheses remained true, we could compute the FDP using D. This process was again repeated for $r = 5000$ for each of the different values of σ_Z^2. The estimate for $P(\overline{FDP} < FDP)$ was computed as follows:

$$\widehat{P(\overline{FDP} < FDP)} = \frac{\#\{k : \overline{FDP_k} < FDP_k\}}{r} \tag{15}$$

Here k indicates iteration k. Since $\#\{k : \overline{FDP_k} < FDP_k\} \sim Bin(P(\overline{FDP} < FDP), r)$, we may again use the normal approximation to the binomial to construct confidence intervals for the estimate. The results are shown below in Table 9.

Table 9: Estimates as function of different dependece

ρ_{ctrl}	$\widehat{P(\overline{FDP} < FDP)}$
0.000	0.010 ± 0.003
0.286	0.029 ± 0.005
0.444	0.036 ± 0.005
0.545	0.038 ± 0.005
0.615	0.041 ± 0.005
0.667	0.043 ± 0.006

Clearly $P(\overline{FDP} < FDP) < \alpha$ still holds. Hence it may be concluded the $(1-\alpha)$-confidence upper bound estimation seems to be robust against heterogeneous dependece between cases and controls. Perhaps for extremely high levels of correlation between the covariates of the controls, $P(\overline{FDP} < FDP)$ would slightly exceed α. Yet it is reasonable to assume such extremely different correlations between groups would typically not be found in practice.

4.2.6 SAM: Heteroscedasticity

In the sixth and final simulation setting we attempted to examine the robustness of the $(1-\alpha)$-confidence upper bound estimation using SAM under heteroscedasticity. We used the same procedure as in the previous simulation setting to compute the required metrics. The data generating process was identical to that of the second simulation setting. The results are shown in Table 10.

Table 10: Estimates as function of increasing heteroscedasticity

σ^2_{ctrl}	$P(\widehat{\overline{FDP} < FDP})$
1.0	0.011 ± 0.003
1.8	0.019 ± 0.004
2.6	0.040 ± 0.005
3.4	$\mathbf{0.055} \pm 0.006$
4.2	$\mathbf{0.072} \pm 0.007$
5.0	$\mathbf{0.099} \pm 0.008$

$P(\overline{FDP} < FDP) < \alpha$ is no longer valid for higher values of σ^2_{ctrl}. Hence it may be concluded the $(1-\alpha)$- confidence upper bound estimation does not seem to be robust in case of heteroscedasticity between cases and controls.

5 Data Illustration

We apply the methods on real data to further illustrate the performance of the methods. We consider the widely used leukemia gene expression dataset from Golub et al. (1999). This dataset contains the expression of 3051 genes on 38 patient samples. The patient samples are divided over two groups. The first group of patients suffer from acute lymphoblastic leukemia (ALL), whereas the second group has acute myeloid leukemia (AML). The groups are not balanced, as $n_{ALL} = 27$, and $n_{AML} = 11$.

5.1 Analysis of heteroscedasticity

Before applying maxT and SAM to the data, we consider potential different variance structures in the data between the groups. Since the simulations showed that dependence is of little concern to FWER control offered maxT and FDP estimation by SAM, we disregard an analysis on potential dependence between genes within groups. On the other hand, the simulations clearly showed heteroscedasticity to be a problem for the considered methods (see Tables 5, 6, 7, 8 and 10).

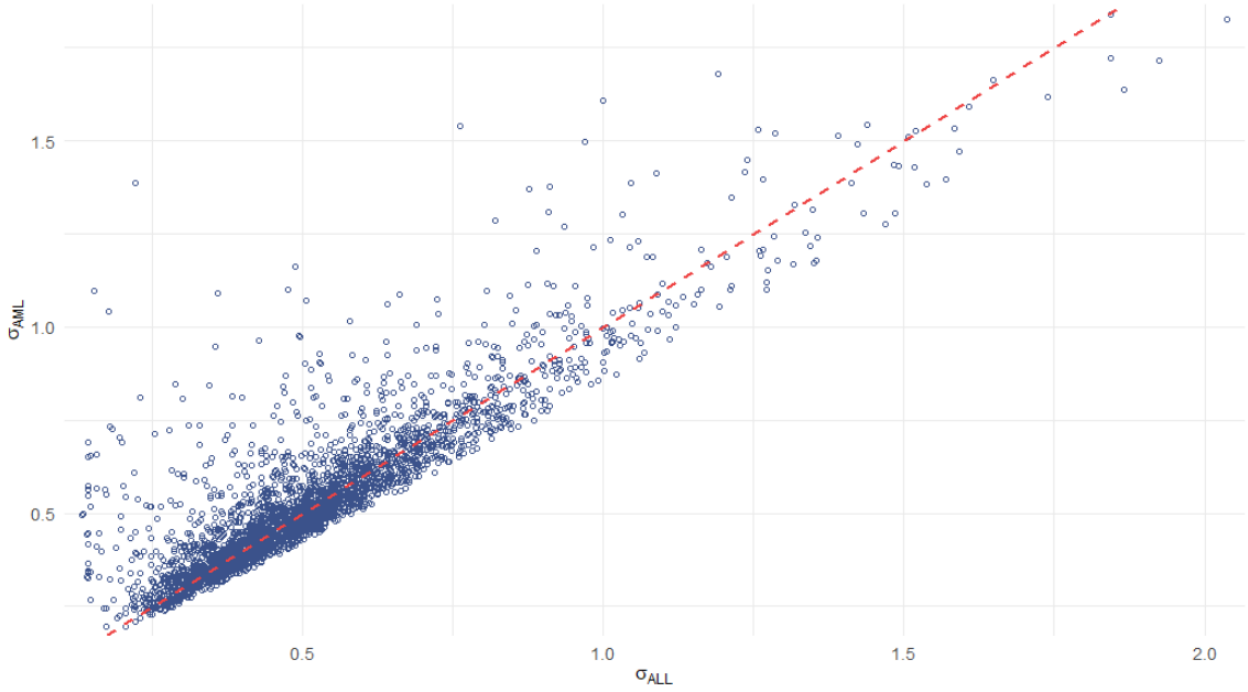

Figure 3: *Scatterplot of within group standard deviations of all genes. Dotted red line shows exact diagonal.*

Figure 3 shows a scatterplot of the standard deviation separated by group of all m gene expressions. The more the points are centered around the red diagonal line, the more individual genes are similar in terms of variance separated by group.

Clearly the AML group has a few dozen more outlying genes compared to the ALL group. This can be attributed to the low sample size of $n_{AML} = 11$, which is not a lot to estimate the standard deviation of a gene with. Nevertheless the bulk of the genes are centered round the diagonal. Thus there might be some very minor heteroscedasticity, but we expect this not to have severe consequences when applying the methods on the data.

5.2 Application to the data

We computed the raw p-values for the leukemia data again using a simple permutation test per hypothesis, as explained in Section 2.1, using the test statistic defined in equation (12). We again set the level of significance to $\alpha = 0.05$. The number of permutations was set to $B = 305100$ in order to keep FWER control by Holm valid. Remarkably $R = 1061$ using the test statistic shown in (12). Considering $m = 3051$, there appears to be a lot of signal in the data. In order to correct for the inevitable presence of Type I errors, we apply maxT, Holm and SAM to the data, and compare the methods. Using the same test statistic, $R_{single} = 94$, $R_{seq} = 98$ and $R_{Holm} = 90$. As expected the number of rejected hypotheses is drastically reduced. Holm and maxT reject a relatively similar amount of hypotheses, as also reflected by their respective FWER estimates seen in almost all of the simulations. Using the Welch t-test statistic, we compute $R = 1052$ using raw p-values, $R_{single}^{Welch} = 92$, $R_{seq}^{Welch} = 98$ and $R_{Holm}^{Welch} = 80$. Welch being less liberal for some procedures may be attributed to the skew in sample size, and slight heteroscedasticity as seen in Figure 3, causing the test statistic to be more appropriate for the data compared to the two-sample t-test statistic.

Recall that SAM rejects hypotheses based on rejection region D, where D is of the the form $(0, c)$. For the simulations we kept $c = 0.01$ constant. When applying SAM to the data example, we vary c to better compare the number of rejections with maxT and Holm. At their most conservative, maxT and Holm reject at the level $\frac{\alpha}{m}$. Therefore we lower c to this level to compare, and observe how the estimate for the FDP, the $(1 - \alpha)$-confidence upper bound and R behave as function of an increasingly more liberal cutoff c. This is displayed in Figure 4.

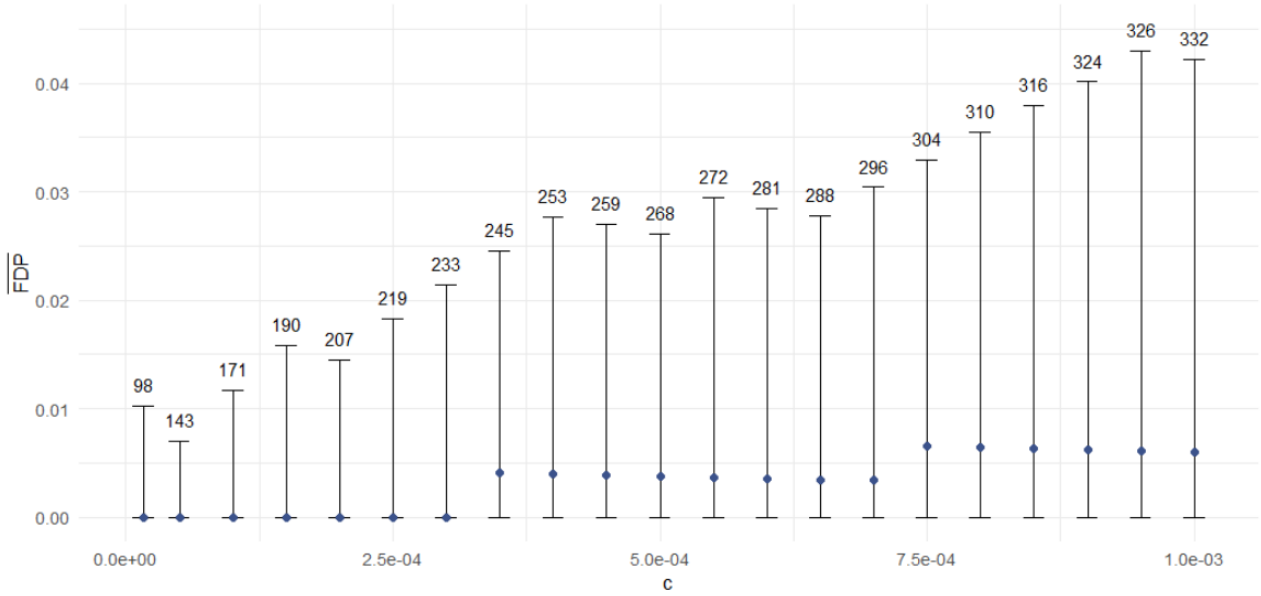

Figure 4: *Points show the SAM estimates of the FDP as function of increasing cutoff (c). Error bars show the estimated* 95% *upper bound for the FDP estimate. Numeric counts above error bars represent the number of rejections R. The test statistic in* (12) *was used, and the number of permutations set to* $B = 100$*. The confidence bounds are not simultaneously valid.*

The first blue dot in Figure 4 represents the FDP estimate produced by SAM at the level $\frac{\alpha}{m}$. Even though the rejection threshold for maxT and Holm presumably does not reach this most conservative level, SAM still rejects a very similar amount hypotheses compared to the previously mentioned methods. The number of rejections by SAM at the Bonferroni level exceeds the number of rejections by Holm due to Holm being based on permutation p-values, whereas the number of rejections for SAM is based on t-test p-values taking into account only the observed data.

Figure 4 shows the number of rejections R increase monotonically with c. A clear trend may also be observed where the $(1-\alpha)$-confidence upper bound increases with a more liberal cutoff. However when looking at the y-axis, it becomes clear the SAM estimate for the FDP is either 0 or close to it, and the $(1-\alpha)$-confidence upper bound for the FDP remains relatively narrow. This practically translates to the following.

Say we were to reject at the level $c = 0.001$, which is the cutoff of the final data point in Figure 4. Here $R = 332$, $\widehat{FDP} = 0.006$ and $\overline{FDP} = 0.042$. Then with 95% confidence we may conclude the true FDP to lie below 0.042, meaning the true number of false positives is with 95% confidence at most $332 \cdot 0.042 \approx 14$, which is remarkably low compared to the total number of rejections made. Naturally this is only the case granted there is no

heteroscedasticity, or another unobserved feature of the data violating assumption B.

6 Discussion

In this thesis we attempted to assess the robustness of two permutation-based multiple testing methods, namely maxT and extension of SAM from Hemerik and Goeman (2018b). This assessment was done by means of simulating case-control specific data configurations, to which the methods were applied. Robustness was gauged through departure (or not) of FWER and FDP estimates from the expected error rates in these particular data configurations. The data configurations in particular violated the prime assumption B underlying both permutation-based multiple testing methods.

To briefly reiterate, according to assumption B we would have to assume that the joint distribution of the part of the data corresponding to the true hypotheses for the cases is the same as for the controls and vice-verse in a case-control setting. In all simulation settings this assumption was violated through either different dependence structures, heteroscedasticity between joint distributions or different marginal distributions. Sometimes this caused the method(s) to break down.

First, we have found evidence that different dependence structures between cases and controls largely do not seem to impact FWER control offered by maxT, nor $(1-\alpha)$-bound estimation for the FDP through SAM, as shown in Tables 4 and 9. However, this was only evaluated for the setting of largest dependence contrast in which all data corresponding to the control group was correlated, whilst keeping all data corresponding to the cases uncorrelated. Even though this could be interpreted as a 'worst case' setting, another less black-and-white dependence structure might result in a different outcome for the robustness of the methods.

Second, it seems conclusive heteroscedasticity between joint distributions causes both maxT and SAM to break down (see Tables 5 and 10). For maxT, the $\widehat{FWER} < \alpha$ is no longer valid as $\widehat{FWER}$ increases dramatically with increasing heteroscedasticity. For SAM the same happens to the required $P(\overline{FDP} < FDP) < \alpha$. Particularly for maxT, slight heteroscedasticity already forms a problem for FWER control. Similar to the dependence setting, this was only evaluated for the setting of largest contrast in variance, where all genes in the control group have a larger variance than all genes in the case group. This structured setting need not be the case as seen in the data example. The methods might be less sensitive to heteroscedasticity under a less black-and-white variance structure.

Third, in an attempt to remedy the problems caused by heteroscedasticity, the use of Welch's t-test statistic did not cause $\widehat{FWER} < \alpha$ to hold. Moreover, differences in sample size between the cases and controls had caused a substantial increase in the estimated FWER (see Tables 5 and 6). Nevertheless the use of Welch's t-test statistic did somewhat reduce the the FWER estimate (see Table 6). Since some heteroscedasticity and sample skew is more conceivable in practice, Welch's t-test statistic almost always seems to be the better choice of test statistic in the context of permutation-based multiple testing.

Fourth, whilst evaluating the performance of FWER control by maxT, the performance of FWER control by the Holm correction was evaluated simultaneously to serve as refer-

ence. In the setting of heteroscedasticity and the use of Welch's t-test statistic, FWER control offered by Holm also broke down. This was unexpected as Janssen (1997) shows permutation tests based on studentized statistics to be asymptotically exact at the level $\alpha = 0.05$. However, Janssen (1997) does not take the multiple testing setting into account. At its most conservative, Holm rejects at the level $\frac{\alpha}{m}$, which is a rejection threshold far deeper into the tail of the null distribution than the considered level of significance. This causes problems for p-value estimation. However, this remains to be evaluated more in depth.

Fifth, different marginal distributions between cases and controls also seems to negatively impact FWER control by maxT, as $\widehat{FWER} < \alpha$ again did not hold. Only the normal-exponential pair was considered. Other combinations of marginals, whilst keeping the first and second moments similar between distributions, might lead to a different conclusion.

To get to these results, some scripting and implementation of algorithms was required. Statistical software in `R` for maxT is quite limited. For the single-step approach to maxT, the *multtest* package by Pollard et al. (2005) is sufficient. However, this package only accounts for a case-control setting. For the complete sequential maxT procedure the *flip* package by Finos, Finos, and Rcpp (2011) is available in all data settings, but unfortunately it turned out too slow for large simulation studies as it not compiled and additionally computes adjusted p-values instead of solely rejecting hypotheses. Therefore the sequential maxT procedure was manually implemented in `R` utilizing the pseudocode found in algorithm block 3. Because this was also not optimized, fewer permutations were used to ease computational demands.

Computational demands overall were a bottleneck in the study, as computation times for individual tables found in Section 4 sometimes spanned more than 24 hours on a 8.00 GB RAM device. In particular, repeatedly computing the $B \times m$ matrix of test statistics, which include a standardization component, required a lot of time. Therefore for future simulation studies regarding this topic, using a higher end device, GPU or server is recommended in order to speed up the simulation process.

Studies potentially building upon the conclusions tied to this thesis could look into the performance of the methods under less black-and-white dependence and heteroscedasticity data configurations. This could also be looked into under different data settings, such as a continuous outcome Y, or paired data instead of case-control data. Additionally, the role of sample sizes, skew in sample sizes, the number of hypotheses and the proportion of true hypotheses could be included as variable parameters in a more extensive study.

7 References

Abdi, H., et al. (2007). Bonferroni and šidák corrections for multiple comparisons. *Encyclopedia of measurement and statistics*, *3*, 103–107.

Benjamini, Y., Drai, D., Elmer, G., Kafkafi, N., & Golani, I. (2001). Controlling the false discovery rate in behavior genetics research. *Behavioural brain research*, *125*(1-2), 279–284.

Benjamini, Y., & 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.

Berry, K. J., Johnston, J. E., & Mielke Jr, P. W. (2014). A chronicle of permutation statistical methods. *Cham: Springer*.

Causeur, D., Friguet, C., Houee-Bigot, M., & Kloareg, M. (2011). Factor analysis for multiple testing (famt): an r package for large-scale significance testing under dependence. *Journal of statistical software*, *40*, 1–19.

Dickhaus, T. (2014). Simultaneous statistical inference. In *With applications in the life sciences.* Springer.

Finos, L., Finos, M. L., & Rcpp, I. (2011). Package 'flip'. https://cran.r-project.org/web/packages/flip/index.html.

Fisher, R. A. (1949). The design of experiments.

Ge, Y., Dudoit, S., & Speed, T. P. (2003). Resampling-based multiple testing for microarray data analysis. *Test*, *12*(1), 1–77.

Goeman, J. J. (2017). De zoekende onderzoeker. Inaugural lecture.

Goeman, J. J., & Solari, A. (2014). Multiple hypothesis testing in genomics. *Statistics in medicine*, *33*(11), 1946–1978.

Golub, T. R., Slonim, D. K., Tamayo, P., Huard, C., Gaasenbeek, M., Mesirov, J. P., ... others (1999). Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *science*, *286*(5439), 531–537.

Hemerik, J., & Goeman, J. (2018a). Exact testing with random permutations. *Test*, *27*(4), 811–825.

Hemerik, J., & Goeman, J. J. (2018b). False discovery proportion estimation by permutations: confidence for significance analysis of microarrays. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *80*(1), 137–155. https://cran.r-project.org/web/packages/confSAM/index.html.

Hemerik, J., & Goeman, J. J. (2021). Another look at the lady tasting tea and differences between permutation tests and randomisation tests. *International Statistical Review*, *89*(2), 367–381.

Hoeffding, W. (1952). The large-sample power of tests based on permutations of observations. *The Annals of Mathematical Statistics*, 169–192.

Holm, S. (1979). A simple sequentially rejective multiple test procedure. *Scandinavian journal of statistics*, 65–70.

Janssen, A. (1997). Studentized permutation tests for non-iid hypotheses and the generalized behrens-fisher problem. *Statistics & probability letters*, *36*(1), 9–21.

Johnson, W. E. (1924). Logic part I. *CUP Archive*.

Lehmann, E. L., Romano, J. P., & Casella, G. (2005). *Testing statistical hypotheses* (Vol. 3). Springer.

Marriott, F. H. (1979). Barnard's monte carlo tests: How many simulations? *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, *28*(1), 75–77.

Menyhart, O., Weltz, B., & Győrffy, B. (2021). Multipletesting. com: a tool for life science researchers for multiple hypothesis testing correction. *PloS one*, *16*(6), e0245824.

Park, H. M. (2008). Hypothesis testing and statistical power of a test. working paper. http://www.indiana.edu/~statmath/stat/all/power/index.html.

Pollard, K. S., Dudoit, S., & van der Laan, M. J. (2005). Multiple testing procedures: the multtest package and applications to genomics. in bioinformatics and computational biology solutions using r and bioconductor. https://www.bioconductor.org/packages/release/bioc/html/multtest.html.

Tusher, V. G., Tibshirani, R., & Chu, G. (2001). Significance analysis of microarrays applied to the ionizing radiation response. *Proceedings of the National Academy of Sciences*, *98*(9), 5116–5121.

Welch, B. L. (1947). The generalization of 'student's'problem when several different population varlances are involved. *Biometrika*, *34*(1-2), 28–35.

Westfall, P. H., & Young, S. S. (1993). *Resampling-based multiple testing: Examples and methods for p-value adjustment* (Vol. 279). John Wiley & Sons.

Appendix A. Simulation scripts

A.1 maxT: Heterogeneous dependence

```
Load packages
```{r}
library(multtest)
library(MASS)
library(ggplot2)
library(reshape2)
```

Basic case control simulation for varying dependence (single-step)
```{r}
simulate <- function(n, ctrl_prop, m, sd_controls, sd_cases, sd_Z, f_prop, f_add, B, test, alpha){

  # Define the number of false hypotheses
  f <- f_prop*m

  # Define groupsizes
  n_ctrl <- n * ctrl_prop
  n_case <- n - n_ctrl

  # Simulate group indicator for all entries (0 = control, 1 = case)
  indicator <- c(rep(0, n_ctrl), rep(1, n_case))

  # Simulate data for cases and controls
  controls <- mvrnorm(n = n_ctrl, mu = rep(0, m), Sigma = diag(m) * (sd_controls)^2, empirical = FALSE)
  cases <- mvrnorm(n = n_case, mu = rep(0, m), Sigma = diag(m) * ((sd_cases)^2) + (sd_Z^2)), empirical = FALSE)

  # Simulate dependency vector or matrix to add to data
  Z <- rnorm(n_ctrl, mean = 0, sd = sd_Z) # Simulate dependency vector to add to data matrix row-wise

  # Add dependency between columns through Z
  controls <- controls + Z

  # Merge cases and controls
  data <- rbind(controls, cases)

  # Check average correlation between columns of only the controls. Cases assumed to have 0 correlation.
  cor <- ((sd_Z)^2)/(sd_controls + (sd_Z)^2)

  # Create f 'false' hypotheses by adding f_add to cases in certain columns
  # mean shift all cases of only false hypotheses, scale with sd_Z
```
```

```
  data[(n_ctrl+1):n, 1:f] <- data[(n_ctrl+1):n, 1:f] + ((f_add/(sqrt(2))) *
      sqrt(sd_controls + ((sd_Z)^2) + sd_cases + ((sd_Z)^2)))

  # Perform MaxT method
  resT <- mt.maxT(X = t(data), classlabel = indicator, test = test, side =
     "abs", B = B)
  rawp <- resT$rawp[order(resT$index)]

  # Perform bonferroni and Holm corrections
  bonf <- p.adjust(rawp, method = "bonferroni")
  holm <- p.adjust(rawp, method = "holm")

  # Create matrix of Type I and Type II errors for each method
  error_mat <- matrix(nrow = 2, ncol = 3)

  error_mat[1,1] <- sum(bonf[(f+1):m] <= 0.05)
  error_mat[2,1] <- sum(bonf[1:f] > 0.05)

  error_mat[1,2] <- sum(holm[(f+1):m] <= 0.05)
  error_mat[2,2] <- sum(holm[1:f] > 0.05)

  error_mat[1,3] <- sum(resT$index > f & resT$adjp <= 0.05)
  error_mat[2,3] <- sum(resT$index <= f & resT$adjp > 0.05)

  # Return these values
  return(list(error_mat, cor))
}

# Set basic simulation values
seed <- 425
r <- 1000
```

Simulate different dependence structure between cases and controls.

```{r}
set.seed(seed)
sd_vec <- sqrt(seq(0, 2, 0.4))
sim_list_1 <- vector("list", length(sd_vec))
fwer_sim1_bonf <- vector(mode = "numeric", length = length(sd_vec))
type_2_sim1_bonf <- vector(mode = "numeric", length = length(sd_vec))
fwer_sim1_holm <- vector(mode = "numeric", length = length(sd_vec))
type_2_sim1_holm <- vector(mode = "numeric", length = length(sd_vec))
fwer_sim1_maxT <- vector(mode = "numeric", length = length(sd_vec))
type_2_sim1_maxT <- vector(mode = "numeric", length = length(sd_vec))
mean_cor <- vector(mode = "numeric", length = length(sd_vec))

# Simulate with different sd_Z for two sample t-statistic
for(i in 1:length(sd_vec)){
  set.seed(seed)
```

```
  sim_list_1[[i]] <- replicate(r, simulate(n = 20, ctrl_prop = 0.5, m =
      500, sd_controls = 1, sd_cases = 1, sd_Z = sd_vec[i], f_prop = 0.1, f_
      add = 2, B = 20000, test = "t.equalvar", alpha = 0.05))
}

# Compute FWER estimate, type 2 errors and (empirical) correlation
for(i in 1:length(sd_vec)){
  for(j in 1:r){
    fwer_sim1_bonf[i] <- fwer_sim1_bonf[i] + (sim_list_1[[i]][c(TRUE, FALSE
        )][[j]][1, 1] >= 1)
    type_2_sim1_bonf[i] <- type_2_sim1_bonf[i] + sim_list_1[[i]][c(TRUE,
        FALSE)][[j]][2, 1]
    fwer_sim1_holm[i] <- fwer_sim1_holm[i] + (sim_list_1[[i]][c(TRUE, FALSE
        )][[j]][1, 2] >= 1)
    type_2_sim1_holm[i] <- type_2_sim1_holm[i] + sim_list_1[[i]][c(TRUE,
        FALSE)][[j]][2, 2]
    fwer_sim1_maxT[i] <- fwer_sim1_maxT[i] + (sim_list_1[[i]][c(TRUE, FALSE
        )][[j]][1, 3] >= 1)
    type_2_sim1_maxT[i] <- type_2_sim1_maxT[i] + sim_list_1[[i]][c(TRUE,
        FALSE)][[j]][2, 3]
    mean_cor[i] <- mean_cor[i] + sim_list_1[[i]][c(FALSE, TRUE)][[j]]
  }
}

fwer_bonf <- fwer_sim1_bonf/r
type_2_bonf <- type_2_sim1_bonf/(r*50)
fwer_holm <- fwer_sim1_holm/r
type_2_holm <- type_2_sim1_holm/(r*50)
fwer_maxT <- fwer_sim1_maxT/r
type_2_maxT <- type_2_sim1_maxT/(r*50)
mean_cor <- mean_cor/r

fwer_bonf
type_2_bonf
fwer_holm
type_2_holm
fwer_maxT
type_2_maxT
mean_cor
```

```
Basic case control simulation for varying dependence (sequential)
```{r}
simulate_seq <- function(n, ctrl_prop, m, sd_controls, sd_cases, sd_Z, f_
    prop, f_add, B, alpha){

  # Define the number of false hypotheses
  f <- f_prop*m

  # Define groupsizes
```
```

```
129  n_ctrl <- n * ctrl_prop
130  n_case <- n - n_ctrl
131
132  # Simulate group indicator for all entries (0 = control, 1 = case)
133  indicator <- c(rep(0, n_ctrl), rep(1, n_case))
134
135  # Simulate data for cases and controls
136  controls <- mvrnorm(n = n_ctrl, mu = rep(0, m), Sigma = diag(m) * (sd_
         controls)^2, empirical = FALSE)
137  cases <- mvrnorm(n = n_case, mu = rep(0, m), Sigma = diag(m) * ((sd_cases
         ^2) + (sd_Z^2)), empirical = FALSE)
138
139  # Simulate dependency vector or matrix to add to data
140  Z <- rnorm(n_ctrl, mean = 0, sd = sd_Z) # Simulate dependency vector to
         add to data matrix row-wise
141
142  # Add dependency between columns through Z
143  controls <- controls + Z
144
145  # Merge cases and controls
146  data <- rbind(controls, cases)
147
148  # Check average correlation between columns of only the controls. Cases
         assumed to have 0 correlation.
149  cor <- ((sd_Z)^2)/(sd_controls + (sd_Z)^2)
150
151  # Create f 'false' hypotheses by adding f_add to cases in certain columns
152  # mean shift all cases of only false hypotheses, scale with sd_Z
153  data[(n_ctrl+1):n, 1:f] <- data[(n_ctrl+1):n, 1:f] + ((f_add/(sqrt(2))) *
          sqrt(sd_controls + ((sd_Z)^2) + sd_cases + ((sd_Z)^2)))
154
155  # Create indicator for each permutation
156  y <- t(replicate(B, sample(indicator, size = n, replace = FALSE)))
157  y[1, ] <- indicator  #add identity permutation
158
159  # Permutation matrix of test statistics
160  t_mat <- matrix(nrow = B, ncol = m)
161  for(b in 1:B){
162    for(j in 1:m){
163      x1 <- data[y[b, ] == 0, j]
164      x2 <- data[y[b, ] == 1, j]
165      n1 <- n_ctrl
166      n2 <- n_case
167      m1 <- mean(x1)
168      m2 <- mean(x2)
169
170      va1 <- ((n1 - 1) * var(x1))
171      va2 <- ((n2 - 1) * var(x2))
172      sp <- sqrt((va1 + va2)/(n1 + n2 - 2))
173
```

```
      t_mat[b,j] <- abs((m1 - m2)/(sp * sqrt((1/n1) + (1/n2))))
    }
  }

  # Initialize rejection vectors
  R <- 1
  R_new <- vector(length = 0)

  # Start sequential rejection procedure
  while(setequal(R, R_new) == FALSE){

    # Update rejection vector
    R <- R_new

    # Get rejection quantile
    Tmax <- apply(t_mat, 1, max, na.rm = TRUE)
    q <- sort(Tmax)[(1 - alpha) * B]

    # Select set of hypotheses to consider
    hypotheses <- 1:m
    if(length(R) > 0){
      hypotheses <- hypotheses[-R]
    }

    # Reject hypotheses based on quantile
    for(j in hypotheses){
      if(t_mat[1, j] > q){
        t_mat[, j] <- NA
        R_new <- c(R_new, j)
      }
    }
  }

  # Return rejections relative to index
  error_vec <- vector(length = 2)

  error_vec[1] <- sum(R > f)
  error_vec[2] <- f - sum(R <= f)

  # Return these values
  return(list(error_vec))
}

# Set basic simulation values
seed <- 425
r <- 5000
```

Simulate different dependence structure between cases and controls.

```{r}
```

```
set.seed(seed)
sd_vec <- sqrt(seq(0, 2, 0.4))
sim_list_1_seq <- vector("list", length(sd_vec))
type_1 <- vector(mode = "numeric", length = length(sd_vec))
type_2 <- vector(mode = "numeric", length = length(sd_vec))

# Simulate with different sd_Z for two sample t-statistic
set.seed(seed)
for(i in 1:length(sd_vec)){
  set.seed(seed)
  sim_list_1_seq[[i]] <- replicate(r, simulate_seq(n = 20, ctrl_prop = 0.5,
       m = 500, sd_controls = 1, sd_cases = 1, sd_Z = sd_vec[i], f_prop =
      0.1, f_add = 2, B = 20, alpha = 0.05))
}

# Compute FWER estimate and type 2 errors
for(i in 1:length(sd_vec)){
  for(j in 1:r){
    type_1[i] <- type_1[i] + sim_list_1_seq[[i]][[j]][1]
    type_2[i] <- type_2[i] + sim_list_1_seq[[i]][[j]][2]
  }
}

fwer <- type_1/r
power <- 1 - type_2/(r*50)

fwer
power

# Get confidence bounds for sequential maxT
1.96*sqrt(((fwer*(1-fwer))/r))
```

A.2 maxT: Heteroscedasticity

```
Load packages
```{r}
library(multtest)
library(MASS)
library(ggplot2)
library(reshape2)
```

Basic case control simulation single-step for heteroscedasticity
```{r}
simulate <- function(n, ctrl_prop, m, sd_controls, sd_cases, f_prop, f_add, B, test, alpha){

  # Define the number of false hypotheses
  f <- f_prop*m

  # Define groupsizes
  n_ctrl <- n * ctrl_prop
  n_case <- n - n_ctrl

  # Simulate group indicator for all entries (0 = control, 1 = case)
  indicator <- c(rep(0, n_ctrl), rep(1, n_case))

  # Simulate data for cases and controls
  controls <- mvrnorm(n = n_ctrl, mu = rep(0, m), Sigma = diag(m) * (sd_controls)^2, empirical = FALSE)
  cases <- mvrnorm(n = n_case, mu = rep(0, m), Sigma = diag(m) * (sd_cases)^2, empirical = FALSE)

  # Merge cases and controls
  data <- rbind(controls, cases)

  # Check average correlation between columns
  cor <- cor(data[1:n_ctrl, ])
  mean_cor <- mean(cor)

  # Create f 'false' hypotheses by adding f_add to cases in certain columns
  # mean shift all cases of only false hypotheses
  data[(n_ctrl+1):n, 1:f] <- data[(n_ctrl+1):n, 1:f] + ((f_add/sqrt(2)) * sqrt((sd_cases)^2 + (sd_controls)^2))

  # Perform MaxT method
  resT <- mt.maxT(X = t(data), classlabel = indicator, test = test, side = "abs", B = B)
  rawp <- resT$rawp[order(resT$index)]

  # Perform bonferroni and Holm corrections
  bonf <- p.adjust(rawp, method = "bonferroni")
```
```

```
  holm <- p.adjust(rawp, method = "holm")

  # Create matrix of Type I and Type II errors for each method
  error_mat <- matrix(nrow = 2, ncol = 3)

  error_mat[1,1] <- sum(bonf[(f+1):m] <= 0.05)
  error_mat[2,1] <- sum(bonf[1:f] > 0.05)

  error_mat[1,2] <- sum(holm[(f+1):m] <= 0.05)
  error_mat[2,2] <- sum(holm[1:f] > 0.05)

  error_mat[1,3] <- sum(resT$index > f & resT$adjp <= 0.05)
  error_mat[2,3] <- sum(resT$index <= f & resT$adjp > 0.05)

  # Return these values
  return(list(error_mat, mean_cor))
}

# Set basic simulation values
seed <- 425
r <- 5000
```

Simulate heteroscedasticity setting, higher variance for controls

```{r}
set.seed(seed)
sd_vec <- sqrt(seq(1, 5, 0.8))
sim_list_1 <- vector("list", length(sd_vec))
fwer_sim1_bonf <- vector(mode = "numeric", length = length(sd_vec))
type_2_sim1_bonf <- vector(mode = "numeric", length = length(sd_vec))
fwer_sim1_holm <- vector(mode = "numeric", length = length(sd_vec))
type_2_sim1_holm <- vector(mode = "numeric", length = length(sd_vec))
fwer_sim1_maxT <- vector(mode = "numeric", length = length(sd_vec))
type_2_sim1_maxT <- vector(mode = "numeric", length = length(sd_vec))
mean_cor <- vector(mode = "numeric", length = length(sd_vec))

# Simulate with different heteroscedasticity for two sample t statistic
for(i in 1:length(sd_vec)){
  set.seed(seed)
  sim_list_1[[i]] <- replicate(r, simulate(n = 20, ctrl_prop = 0.5, m =
      500, sd_controls = sd_vec[i], sd_cases = 1, f_prop = 0.1, f_add = 2, B
       = 20000, test = "t.equalvar", alpha = 0.05))
}

# compute FWER estimate, type 2 errors and empirical correlation
for(i in 1:length(sd_vec)){
  for(j in 1:r){
    fwer_sim1_bonf[i] <- fwer_sim1_bonf[i] + (sim_list_1[[i]][c(TRUE, FALSE
       )][[j]][1, 1] >= 1)
```

```
    type_2_sim1_bonf[i] <- type_2_sim1_bonf[i] + sim_list_1[[i]][c(TRUE,
        FALSE)][[j]][2, 1]
    fwer_sim1_holm[i] <- fwer_sim1_holm[i] + (sim_list_1[[i]][c(TRUE, FALSE
        )][[j]][1, 2] >= 1)
    type_2_sim1_holm[i] <- type_2_sim1_holm[i] + sim_list_1[[i]][c(TRUE,
        FALSE)][[j]][2, 2]
    fwer_sim1_maxT[i] <- fwer_sim1_maxT[i] + (sim_list_1[[i]][c(TRUE, FALSE
        )][[j]][1, 3] >= 1)
    type_2_sim1_maxT[i] <- type_2_sim1_maxT[i] + sim_list_1[[i]][c(TRUE,
        FALSE)][[j]][2, 3]
    mean_cor[i] <- mean_cor[i] + sim_list_1[[i]][c(FALSE, TRUE)][[j]]
  }
}

fwer_bonf <- fwer_sim1_bonf/r
type_2_bonf <- type_2_sim1_bonf/(r*50)
fwer_holm <- fwer_sim1_holm/r
type_2_holm <- type_2_sim1_holm/(r*50)
fwer_maxT <- fwer_sim1_maxT/r
type_2_maxT <- type_2_sim1_maxT/(r*50)
mean_cor <- mean_cor/r

fwer_bonf
type_2_bonf
fwer_holm
type_2_holm
fwer_maxT
type_2_maxT
mean_cor
```

```
Display fwer with confidence bounds and power estimates
```{r}
fwer_maxT
fwer_holm
1.96*sqrt(((fwer_maxT*(1-fwer_maxT))/r))
1.96*sqrt(((fwer_holm*(1-fwer_holm))/r))
1 - type_2_maxT
1 - type_2_holm
```

```
Basic case control simulation for varying heteroscedasticity (sequential)
```{r}
simulate_seq <- function(n, ctrl_prop, m, sd_controls, sd_cases, f_prop, f_
    add, B, alpha){

  # Define the number of false hypotheses
  f <- f_prop*m

  # Define groupsizes
```

```
134  n_ctrl <- n * ctrl_prop
135  n_case <- n - n_ctrl
136
137  # Simulate group indicator for all entries (0 = control, 1 = case)
138  indicator <- c(rep(0, n_ctrl), rep(1, n_case))
139
140  # Simulate data for cases and controls
141  controls <- mvrnorm(n = n_ctrl, mu = rep(0, m), Sigma = diag(m) * (sd_
         controls)^2, empirical = FALSE)
142  cases <- mvrnorm(n = n_case, mu = rep(0, m), Sigma = diag(m) * (sd_cases)
         ^2, empirical = FALSE)
143
144  # Merge cases and controls
145  data <- rbind(controls, cases)
146
147  # Create f 'false' hypotheses by adding f_add to cases in certain columns
148  # mean shift all cases of only false hypotheses
149  data[(n_ctrl+1):n, 1:f] <- data[(n_ctrl+1):n, 1:f] + ((f_add/sqrt(2)) *
         sqrt((sd_cases)^2 + (sd_controls)^2))
150
151  # Create indicator for each permutation
152  y <- t(replicate(B, sample(indicator, size = n, replace = FALSE)))
153  y[1, ] <- indicator  #add identity permutation
154
155  # Permutation matrix of test statistics
156  t_mat <- matrix(nrow = B, ncol = m)
157  for(b in 1:B){
158    for(j in 1:m){
159      x1 <- data[y[b, ] == 0, j]
160      x2 <- data[y[b, ] == 1, j]
161      n1 <- n_ctrl
162      n2 <- n_case
163      m1 <- mean(x1)
164      m2 <- mean(x2)
165
166      va1 <- ((n1 - 1) * var(x1))
167      va2 <- ((n2 - 1) * var(x2))
168      sp <- sqrt((va1 + va2)/(n1 + n2 - 2))
169
170      t_mat[b,j] <- abs((m1 - m2)/(sp * sqrt((1/n1) + (1/n2))))
171    }
172  }
173
174  # Initialize rejection vectors
175  R <- 1
176  R_new <- vector(length = 0)
177
178  # Start sequential rejection procedure
179  while(setequal(R, R_new) == FALSE){
180
```

```
    # Update rejection vector
    R <- R_new

    # Get rejection quantile
    Tmax <- apply(t_mat, 1, max, na.rm = TRUE)
    q <- sort(Tmax)[(1 - alpha) * B]

    # Select set of hypotheses to consider
    hypotheses <- 1:m
    if(length(R) > 0){
      hypotheses <- hypotheses[-R]
    }

    # Reject hypotheses based on quantile
    for(j in hypotheses){
      if(t_mat[1, j] > q){
        t_mat[, j] <- NA
        R_new <- c(R_new, j)
      }
    }
  }
  R

  # Return rejections relative to index
  error_vec <- vector(length = 2)

  error_vec[1] <- sum(R > f)
  error_vec[2] <- f - sum(R <= f)

  # Return these values
  return(list(error_vec))
}

# Set basic simulation values
seed <- 425
r <- 5000
```

Simulate different dependence structure between cases and controls.

```{r}
set.seed(seed)
sd_vec <- sqrt(seq(1, 5, 0.8))
sim_list_1_seq <- vector("list", length(sd_vec))
type_1 <- vector(mode = "numeric", length = length(sd_vec))
type_2 <- vector(mode = "numeric", length = length(sd_vec))

# Simulate with different heteroscedasticity for two sample t statistic
set.seed(seed)
for(i in 1:length(sd_vec)){
  set.seed(seed)
```

```
231   sim_list_1_seq[[i]] <- replicate(r, simulate_seq(n = 20, ctrl_prop = 0.5,
          m = 500, sd_controls = sd_vec[i], sd_cases = 1, f_prop = 0.1, f_add =
          2, B = 20, alpha = 0.05))
232 }
233
234 # Compute FWER estimate and count type 2 errors
235 for(i in 1:length(sd_vec)){
236   for(j in 1:r){
237     type_1[i] <- type_1[i] + sim_list_1_seq[[i]][[j]][1]
238     type_2[i] <- type_2[i] + sim_list_1_seq[[i]][[j]][2]
239   }
240 }
241
242 fwer <- type_1/r
243 power <- 1 - type_2/(r*50)
244
245 fwer
246 power
247
248 # Get confidence bounds for sequential maxT
249 1.96*sqrt(((fwer*(1-fwer))/r))
250 ```
```

A.3 maxT: Heteroscedasticity (Welch)

```
Load packages
```{r}
library(multtest)
library(MASS)
library(ggplot2)
library(reshape2)
```

Basic case control simulation for single-step under heteroscedasticity
```{r}
simulate <- function(n, ctrl_prop, m, sd_controls, sd_cases, f_prop, f_add, B, test, alpha){

  # Define the number of false hypotheses
  f <- f_prop*m

  # Define groupsizes
  n_ctrl <- n * ctrl_prop
  n_case <- n - n_ctrl

  # Simulate group indicator for all entries (0 = control, 1 = case)
  indicator <- c(rep(0, n_ctrl), rep(1, n_case))

  # Simulate data for cases and controls
  controls <- mvrnorm(n = n_ctrl, mu = rep(0, m), Sigma = diag(m) * (sd_controls)^2, empirical = FALSE)
  cases <- mvrnorm(n = n_case, mu = rep(0, m), Sigma = diag(m) * (sd_cases)^2, empirical = FALSE)

  # Merge cases and controls
  data <- rbind(controls, cases)

  # Check average correlation between columns
  cor <- cor(data[1:n_ctrl, ])
  mean_cor <- mean(cor)

  # Create f 'false' hypotheses by adding f_add to cases in certain columns
  # mean shift all cases of only false hypotheses
  data[(n_ctrl+1):n, 1:f] <- data[(n_ctrl+1):n, 1:f] + ((f_add/sqrt(2)) * sqrt((sd_cases)^2 + (sd_controls)^2))

  # Perform MaxT method
  resT <- mt.maxT(X = t(data), classlabel = indicator, test = test, side = "abs", B = B)
  rawp <- resT$rawp[order(resT$index)]

  # Perform bonferroni and Holm corrections
  bonf <- p.adjust(rawp, method = "bonferroni")
```
```

```
  holm <- p.adjust(rawp, method = "holm")

  # Create matrix of Type I and Type II errors for each method
  error_mat <- matrix(nrow = 2, ncol = 3)

  error_mat[1,1] <- sum(bonf[(f+1):m] <= 0.05)
  error_mat[2,1] <- sum(bonf[1:f] > 0.05)

  error_mat[1,2] <- sum(holm[(f+1):m] <= 0.05)
  error_mat[2,2] <- sum(holm[1:f] > 0.05)

  error_mat[1,3] <- sum(resT$index > f & resT$adjp <= 0.05)
  error_mat[2,3] <- sum(resT$index <= f & resT$adjp > 0.05)

  # Return these values
  return(list(error_mat, mean_cor))
}

# Set basic simulation values
seed <- 425
r <- 5000
```

```
Simulate heteroscedasticity setting, higher variance for controls (Welch)
```{r}
set.seed(seed)
sd_vec <- sqrt(seq(1, 5, 0.8))
sim_list_1 <- vector("list", length(sd_vec))
fwer_sim1_bonf <- vector(mode = "numeric", length = length(sd_vec))
type_2_sim1_bonf <- vector(mode = "numeric", length = length(sd_vec))
fwer_sim1_holm <- vector(mode = "numeric", length = length(sd_vec))
type_2_sim1_holm <- vector(mode = "numeric", length = length(sd_vec))
fwer_sim1_maxT <- vector(mode = "numeric", length = length(sd_vec))
type_2_sim1_maxT <- vector(mode = "numeric", length = length(sd_vec))
mean_cor <- vector(mode = "numeric", length = length(sd_vec))

# Simulate with increasing heteroscedasticity with Welch t statistic
for(i in 1:length(sd_vec)){
  set.seed(seed)
  sim_list_1[[i]] <- replicate(r, simulate(n = 20, ctrl_prop = 0.4, m =
      500, sd_controls = sd_vec[i], sd_cases = 1, f_prop = 0.1, f_add = 2, B
       = 20000, test = "t", alpha = 0.05))
}

# Compute FWER estimate, type 2 error count and empirical correlation
for(i in 1:length(sd_vec)){
  for(j in 1:r){
    fwer_sim1_bonf[i] <- fwer_sim1_bonf[i] + (sim_list_1[[i]][c(TRUE, FALSE
        )][[j]][1, 1] >= 1)
```
```

```
    type_2_sim1_bonf[i] <- type_2_sim1_bonf[i] + sim_list_1[[i]][c(TRUE,
        FALSE)][[j]][2, 1]
    fwer_sim1_holm[i] <- fwer_sim1_holm[i] + (sim_list_1[[i]][c(TRUE, FALSE
        )][[j]][1, 2] >= 1)
    type_2_sim1_holm[i] <- type_2_sim1_holm[i] + sim_list_1[[i]][c(TRUE,
        FALSE)][[j]][2, 2]
    fwer_sim1_maxT[i] <- fwer_sim1_maxT[i] + (sim_list_1[[i]][c(TRUE, FALSE
        )][[j]][1, 3] >= 1)
    type_2_sim1_maxT[i] <- type_2_sim1_maxT[i] + sim_list_1[[i]][c(TRUE,
        FALSE)][[j]][2, 3]
    mean_cor[i] <- mean_cor[i] + sim_list_1[[i]][c(FALSE, TRUE)][[j]]
  }
}

fwer_bonf <- fwer_sim1_bonf/r
type_2_bonf <- type_2_sim1_bonf/(r*50)
fwer_holm <- fwer_sim1_holm/r
type_2_holm <- type_2_sim1_holm/(r*50)
fwer_maxT <- fwer_sim1_maxT/r
type_2_maxT <- type_2_sim1_maxT/(r*50)
mean_cor <- mean_cor/r

fwer_bonf
type_2_bonf
fwer_holm
type_2_holm
fwer_maxT
type_2_maxT
mean_cor
```

Simulate heteroscedasticity setting, higher variance for controls (normal two sample t-test)

```{r}
set.seed(seed)
sd_vec <- sqrt(seq(1, 5, 0.8))
sim_list_2 <- vector("list", length(sd_vec))
fwer_sim2_bonf <- vector(mode = "numeric", length = length(sd_vec))
type_2_sim2_bonf <- vector(mode = "numeric", length = length(sd_vec))
fwer_sim2_holm <- vector(mode = "numeric", length = length(sd_vec))
type_2_sim2_holm <- vector(mode = "numeric", length = length(sd_vec))
fwer_sim2_maxT <- vector(mode = "numeric", length = length(sd_vec))
type_2_sim2_maxT <- vector(mode = "numeric", length = length(sd_vec))
mean_cor <- vector(mode = "numeric", length = length(sd_vec))

# # Simulate with increasing heteroscedasticity with two sample t-statistic
for(i in 1:length(sd_vec)){
  set.seed(seed)
  sim_list_2[[i]] <- replicate(r, simulate(n = 20, ctrl_prop = 0.4, m =
      500, sd_controls = sd_vec[i], sd_cases = 1, f_prop = 0.1, f_add = 2, B
```

```
        = 20000, test = "t.equalvar", alpha = 0.05))
}

# Compute FWER estimate, type 2 error count and empirical correlation
for(i in 1:length(sd_vec)){
  for(j in 1:r){
    fwer_sim2_bonf[i] <- fwer_sim2_bonf[i] + (sim_list_2[[i]][c(TRUE, FALSE
        )][[j]][1, 1] >= 1)
    type_2_sim2_bonf[i] <- type_2_sim2_bonf[i] + sim_list_2[[i]][c(TRUE,
        FALSE)][[j]][2, 1]
    fwer_sim2_holm[i] <- fwer_sim2_holm[i] + (sim_list_2[[i]][c(TRUE, FALSE
        )][[j]][1, 2] >= 1)
    type_2_sim2_holm[i] <- type_2_sim2_holm[i] + sim_list_2[[i]][c(TRUE,
        FALSE)][[j]][2, 2]
    fwer_sim2_maxT[i] <- fwer_sim2_maxT[i] + (sim_list_2[[i]][c(TRUE, FALSE
        )][[j]][1, 3] >= 1)
    type_2_sim2_maxT[i] <- type_2_sim2_maxT[i] + sim_list_2[[i]][c(TRUE,
        FALSE)][[j]][2, 3]
    mean_cor[i] <- mean_cor[i] + sim_list_2[[i]][c(FALSE, TRUE)][[j]]
  }
}

fwer_bonf_sim2 <- fwer_sim2_bonf/r
type_2_bonf_sim2 <- type_2_sim2_bonf/(r*50)
fwer_holm_sim2 <- fwer_sim2_holm/r
type_2_holm_sim2 <- type_2_sim2_holm/(r*50)
fwer_maxT_sim2 <- fwer_sim2_maxT/r
type_2_maxT_sim2 <- type_2_sim2_maxT/(r*50)
mean_cor_sim2 <- mean_cor/r

fwer_bonf_sim2
type_2_bonf_sim2
fwer_holm_sim2
type_2_holm_sim2
fwer_maxT_sim2
type_2_maxT_sim2
mean_cor_sim2
```

```
Display fwer with confidence bounds and power estimates
```{r}
#MaxT single-step

fwer_maxT_sim2
fwer_maxT
1.96*sqrt(((fwer_maxT_sim2*(1-fwer_maxT_sim2))/r))
1.96*sqrt(((fwer_maxT*(1-fwer_maxT))/r))
1 - type_2_maxT_sim2
1 - type_2_maxT

```
```

```
#Holm

fwer_holm_sim2
fwer_holm
1.96*sqrt(((fwer_holm_sim2*(1-fwer_holm_sim2))/r))
1.96*sqrt(((fwer_holm*(1-fwer_holm))/r))
1 - type_2_holm_sim2
1 - type_2_holm
```

Basic case control simulation for varying heteroscedasticity, normal t-test (sequential)

```{r}
simulate_seq <- function(n, ctrl_prop, m, sd_controls, sd_cases, f_prop, f_add, B, alpha){

  # Define the number of false hypotheses
  f <- f_prop*m

  # Define groupsizes
  n_ctrl <- n * ctrl_prop
  n_case <- n - n_ctrl

  # Simulate group indicator for all entries (0 = control, 1 = case)
  indicator <- c(rep(0, n_ctrl), rep(1, n_case))

  # Simulate data for cases and controls
  controls <- mvrnorm(n = n_ctrl, mu = rep(0, m), Sigma = diag(m) * (sd_controls)^2, empirical = FALSE)
  cases <- mvrnorm(n = n_case, mu = rep(0, m), Sigma = diag(m) * (sd_cases)^2, empirical = FALSE)

  # Merge cases and controls
  data <- rbind(controls, cases)

  # Create f 'false' hypotheses by adding f_add to cases in certain columns
  # mean shift all cases of only false hypotheses
  data[(n_ctrl+1):n, 1:f] <- data[(n_ctrl+1):n, 1:f] + ((f_add/sqrt(2)) * sqrt((sd_cases)^2 + (sd_controls)^2))

  # Create indicator for each permutation
  y <- t(replicate(B, sample(indicator, size = n, replace = FALSE)))
  y[1, ] <- indicator  #add identity permutation

  # Permutation matrix of test statistics
  t_mat <- matrix(nrow = B, ncol = m)
  for(b in 1:B){
    for(j in 1:m){
      x1 <- data[y[b, ] == 0, j]
      x2 <- data[y[b, ] == 1, j]
```

```
      n1 <- n_ctrl
      n2 <- n_case
      m1 <- mean(x1)
      m2 <- mean(x2)

      va1 <- ((n1 - 1) * var(x1))
      va2 <- ((n2 - 1) * var(x2))
      sp <- sqrt((va1 + va2)/(n1 + n2 - 2))

      t_mat[b,j] <- abs((m1 - m2)/(sp * sqrt((1/n1) + (1/n2))))
    }
  }

  # Initialize rejection vectors
  R <- 1
  R_new <- vector(length = 0)

  # Start sequential rejection procedure
  while(setequal(R, R_new) == FALSE){

    # Update rejection vector
    R <- R_new

    # Get rejection quantile
    Tmax <- apply(t_mat, 1, max, na.rm = TRUE)
    q <- sort(Tmax)[(1 - alpha) * B]

    # Select set of hypotheses to consider
    hypotheses <- 1:m
    if(length(R) > 0){
      hypotheses <- hypotheses[-R]
    }

    # Reject hypotheses based on quantile
    for(j in hypotheses){
      if(t_mat[1, j] > q){
        t_mat[, j] <- NA
        R_new <- c(R_new, j)
      }
    }
  }
  R

  # Return rejections relative to index
  error_vec <- vector(length = 2)

  error_vec[1] <- sum(R > f)
  error_vec[2] <- f - sum(R <= f)

  # Return these values
```

```
  return(list(error_vec))
}

# Set basic simulation values
seed <- 425
r <- 5000
```

Simulate different dependence structure between cases and controls.

```{r}
set.seed(seed)
sd_vec <- sqrt(seq(1, 5, 0.8))
sim_list_1_seq <- vector("list", length(sd_vec))
type_1 <- vector(mode = "numeric", length = length(sd_vec))
type_2 <- vector(mode = "numeric", length = length(sd_vec))

# Simulate with different heteroscedascitiy for two sample t-test statistic
set.seed(seed)
for(i in 1:length(sd_vec)){
  set.seed(seed)
  sim_list_1_seq[[i]] <- replicate(r, simulate_seq(n = 20, ctrl_prop = 0.4,
      m = 500, sd_controls = sd_vec[i], sd_cases = 1, f_prop = 0.1, f_add =
      2, B = 20, alpha = 0.05))
}

# Compute FWER estimate and Type 2 error count
for(i in 1:length(sd_vec)){
  for(j in 1:r){
    type_1[i] <- type_1[i] + sim_list_1_seq[[i]][[j]][1]
    type_2[i] <- type_2[i] + sim_list_1_seq[[i]][[j]][2]
  }
}

fwer <- type_1/r
power <- 1 - type_2/(r*50)

fwer
power

# Get confidence bounds for sequential maxT
1.96*sqrt(((fwer*(1-fwer))/r))
```

A.4 maxT: Different distributions

```
Load packages
```{r}
library(multtest)
library(MASS)
library(ggplot2)
library(reshape2)
```

Basic case control simulation for different marginal distributions (exponential - normal)
```{r}
simulate_distr <- function(n, ctrl_prop, m, sd_controls, lambda, f_prop, f_add, B, test, alpha){

  # Define the number of false hypotheses
  f <- f_prop*m

  # Define standard deviation of cases
  sd_cases <- sqrt(1/(lambda^2))

  # Define groupsizes
  n_ctrl <- n * ctrl_prop
  n_case <- n - n_ctrl

  # Simulate group indicator for all entries (0 = control, 1 = case)
  indicator <- c(rep(0, n_ctrl), rep(1, n_case))

  # Simulate data for cases and controls
  controls <- mvrnorm(n = n_ctrl, mu = rep(0, m), Sigma = diag(m) * (sd_controls)^2, empirical = FALSE)
  cases <- matrix(rexp((n_case)*m, rate = lambda), nrow = n_case, ncol = m) - (1/lambda)

  # Merge cases and controls
  data <- rbind(controls, cases)

  # Create f 'false' hypotheses by adding f_add to cases in certain columns
  # mean shift all cases of only false hypotheses
  data[(n_ctrl+1):n, 1:f] <- data[(n_ctrl+1):n, 1:f] + ((f_add/sqrt(2)) * sqrt((sd_cases)^2 + (sd_controls)^2))


  # Perform MaxT method
  resT <- mt.maxT(X = t(data), classlabel = indicator, test = test, side = "abs", B = B)
  rawp <- resT$rawp[order(resT$index)]

  # Perform bonferroni and Holm corrections
```
```

```
  bonf <- p.adjust(rawp, method = "bonferroni")
  holm <- p.adjust(rawp, method = "holm")

  # Create matrix of Type I and Type II errors for each method
  error_mat <- matrix(nrow = 2, ncol = 3)

  error_mat[1,1] <- sum(bonf[(f+1):m] <= 0.05)
  error_mat[2,1] <- sum(bonf[1:f] > 0.05)

  error_mat[1,2] <- sum(holm[(f+1):m] <= 0.05)
  error_mat[2,2] <- sum(holm[1:f] > 0.05)

  error_mat[1,3] <- sum(resT$index > f & resT$adjp <= 0.05)
  error_mat[2,3] <- sum(resT$index <= f & resT$adjp > 0.05)

  # Return these values
  return(error_mat)
}

# Set basic simulation values
seed <- 425
r <- 5000
```

```
Simulate for different marginals
```{r}
fwer_sim1_holm <- vector("numeric", length = 1)
type_2_sim1_holm <- vector("numeric", length = 1)
fwer_sim1_maxT <- vector("numeric", length = 1)
type_2_sim1_maxT <- vector("numeric", length = 1)

# Controls are exp, cases are normal
set.seed(seed)
sim_1 <- replicate(r, simulate_distr(n = 20, ctrl_prop = 0.5, m = 500, sd_
    controls = 1, lambda = 1, f_prop = 0.1, f_add = 2, B = 20000, test = "t.
    equalvar", alpha = 0.05))

# Compute FWER estimate and type 2 error count
for(j in 1:r){
    fwer_sim1_holm <- fwer_sim1_holm + (sim_1[1, 2, j] >= 1)
    type_2_sim1_holm <- type_2_sim1_holm + sim_1[2, 2, j]
    fwer_sim1_maxT <- fwer_sim1_maxT + (sim_1[1, 3, j] >= 1)
    type_2_sim1_maxT <- type_2_sim1_maxT + sim_1[2, 3, j]
    }

fwer_sim1_holm <- fwer_sim1_holm/r
fwer_sim1_maxT <- fwer_sim1_maxT/r
type_2_sim1_holm <- type_2_sim1_holm/(r*50)
type_2_sim1_maxT <- type_2_sim1_maxT/(r*50)

```
```

```
fwer_sim1_holm
fwer_sim1_maxT
type_2_sim1_holm
type_2_sim1_maxT

1.96*sqrt(((fwer_sim1_holm*(1-fwer_sim1_holm))/r))
1.96*sqrt(((type_2_sim1_maxT*(1-type_2_sim1_maxT))/r))
1 - type_2_sim1_holm
1 - type_2_sim1_maxT
```

```
Basic case control simulation for different marginal distributions (sequential)
```{r}
simulate_seq_w <- function(n, ctrl_prop, m, sd_controls, lambda, f_prop, f_add, B, alpha){

  # Define the number of false hypotheses
  f <- f_prop*m

  # Define standard deviation of cases
  sd_cases <- sqrt(1/(lambda^2))

  # Define groupsizes
  n_ctrl <- n * ctrl_prop
  n_case <- n - n_ctrl

  # Simulate group indicator for all entries (0 = control, 1 = case)
  indicator <- c(rep(0, n_ctrl), rep(1, n_case))

  # Simulate data for cases and controls
  controls <- mvrnorm(n = n_ctrl, mu = rep(0, m), Sigma = diag(m) * (sd_controls)^2, empirical = FALSE)
  cases <- matrix(rexp((n_case)*m, rate = lambda), nrow = n_case, ncol = m) - (1/lambda)

  # Merge cases and controls
  data <- rbind(controls, cases)

  # Create f 'false' hypotheses by adding f_add to cases in certain columns
  # mean shift all cases of only false hypotheses
  data[(n_ctrl+1):n, 1:f] <- data[(n_ctrl+1):n, 1:f] + ((f_add/sqrt(2)) * sqrt((sd_cases)^2 + (sd_controls)^2))

  # Create indicator for each permutation
  y <- t(replicate(B, sample(indicator, size = n, replace = FALSE)))
  y[1, ] <- indicator  #add identity permutation

  # Permutation matrix of test statistics
  t_mat <- matrix(nrow = B, ncol = m)
```
```

```
136   for(b in 1:B){
137     for(j in 1:m){
138       x1 <- data[y[b, ] == 0, j]
139       x2 <- data[y[b, ] == 1, j]
140       n1 <- n_ctrl
141       n2 <- n_case
142       m1 <- mean(x1)
143       m2 <- mean(x2)
144
145       sd1 <- sd(x1)
146       sd2 <- sd(x2)
147       sdx1 <- sd1/sqrt(n1)
148       sdx2 <- sd2/sqrt(n2)
149
150       t_mat[b,j] <- abs((m1 - m2)/(sqrt((sdx1^2) + (sdx2^2))))
151     }
152   }
153
154   # Initialize rejection vectors
155   R <- 1
156   R_new <- vector(length = 0)
157
158   # Start sequential rejection procedure
159   while(setequal(R, R_new) == FALSE){
160
161     # Update rejection vector
162     R <- R_new
163
164     # Get rejection quantile
165     Tmax <- apply(t_mat, 1, max, na.rm = TRUE)
166     q <- sort(Tmax)[(1 - alpha) * B]
167
168     # Select set of hypotheses to consider
169     hypotheses <- 1:m
170     if(length(R) > 0){
171       hypotheses <- hypotheses[-R]
172     }
173
174     # Reject hypotheses based on quantile
175     for(j in hypotheses){
176       if(t_mat[1, j] > q){
177         t_mat[, j] <- NA
178         R_new <- c(R_new, j)
179       }
180     }
181   }
182   R
183
184   # Return rejections relative to index
185   error_vec <- vector(length = 2)
```

```
186
187   error_vec[1] <- sum(R > f)
188   error_vec[2] <- f - sum(R <= f)
189
190   # Return these values
191   return(list(error_vec))
192 }
193
194 # Set basic simulation values
195 seed <- 425
196 r <- 5000
197 ```
198
199 Simulate for different marginals
200 ```{r}
201 type_1_w <- vector("numeric", length = 1)
202 type_2_w <- vector("numeric", length = 1)
203
204 # Simulate
205 set.seed(seed)
206 sim_1_seq_w <- replicate(r, simulate_seq_w(n = 20, ctrl_prop = 0.5, m =
       500, sd_controls = 1, lambda = 1, f_prop = 0.1, f_add = 2, B = 20, alpha
        = 0.05))
207
208 # Compute FWER estimate and type 2 error count
209 for(j in 1:r){
210   type_1_w <- type_1_w + sim_1_seq_w[[j]][1]
211   type_2_w <- type_2_w + sim_1_seq_w[[j]][2]
212 }
213
214 fwer_w <- type_1_w/r
215 power_w <- 1 - type_2_w/(r*50)
216
217 fwer_w
218 power_w
219
220 # Get confidence bounds for sequential maxT
221 1.96*sqrt(((fwer_w*(1-fwer_w))/r))
222 ```
```

A.5 SAM: Heterogeneous dependence

```
```{r}
library(MASS)
library(confSAM)
```

Case control simulation variable correlation structure
```{r}
simulate <- function(n, ctrl_prop, m, sd_controls, sd_cases, sd_Z, f_prop, f_add, B, cutoff, alpha){

  # Define the number of false hypotheses
  f <- f_prop*m

  # Define groupsizes
  n_ctrl <- n * ctrl_prop
  n_case <- n - n_ctrl

  # Simulate group indicator for all entries (0 = control, 1 = case)
  indicator <- c(rep(0, n_ctrl), rep(1, n_case))

  # Simulate data for cases and controls
  controls <- mvrnorm(n = n_ctrl, mu = rep(0, m), Sigma = diag(m) * (sd_controls)^2, empirical = FALSE)
  cases <- mvrnorm(n = n_case, mu = rep(0, m), Sigma = diag(m) * ((sd_cases^2) + (sd_Z^2)), empirical = FALSE)

  # Simulate dependency vector or matrix to add to data
  Z <- rnorm(n_ctrl, mean = 0, sd = sd_Z) # Simulate dependency vector to add to data matrix row-wise

  # Add dependency between columns through Z
  controls <- controls + Z

  # Merge cases and controls
  data <- rbind(controls, cases)

  # Check average correlation between columns of only the controls. Cases assumed to have 0 correlation.
  # cor <- ((sd_Z)^2)/(sd_controls + (sd_Z)^2)

  # Create f 'false' hypotheses by adding f_add to cases in certain columns
  # mean shift all cases of only false hypotheses, scale with sd_Z
  data[(n_ctrl+1):n, 1:f] <- data[(n_ctrl+1):n, 1:f] + ((f_add/(sqrt(2))) * sqrt(sd_controls + ((sd_Z)^2) + sd_cases + ((sd_Z)^2)))

  # Create indicator for each permutation
  y <- t(replicate(B, sample(indicator, size = n, replace = FALSE)))
  y[1,] <- indicator  #add identity permutation
```
```

```
43
44  # Permutation p-values
45  p_mat <- matrix(nrow = B, ncol = m)
46  for(b in 1:B){
47    for(j in 1:m){
48      x1 <- data[y[b, ] == 0, j]
49      x2 <- data[y[b, ] == 1, j]
50      n1 <- n_ctrl
51      n2 <- n_case
52      m1 <- mean(x1)
53      m2 <- mean(x2)
54
55      va1 <- ((n1 - 1) * var(x1))
56      va2 <- ((n2 - 1) * var(x2))
57      sp <- sqrt((va1 + va2)/(n1 + n2 - 2))
58
59      teststat <- abs((m1 - m2)/(sp * sqrt((1/n1) + (1/n2))))
60      p_mat[b,j] <- 2 * pt(teststat, n1 + n2 - 2, lower = FALSE)
61    }
62  }
63
64  # Permutation p-values
65  # p_mat <- matrix(nrow = B, ncol = m)
66  # for(b in 1:B){
67  #   for(j in 1:m){
68  #     p_mat[b,j] <- t.test(data[y[b, ] == 1, j], data[y[b, ] == 0, j],
        alternative = "two.sided" )$p.value
69  #   }
70  # }
71
72  # Sample pvalues using permutation t-test
73  # p_0 <- vector(length = 0)
74  # for(j in 1:m){
75  #   p_0[j] <- (sum(p_mat[1, j] >= p_mat[2:B, j])+1)/B
76  # }
77
78  # Sample pvalues using t-test
79  # p_0 <- vector(length = m)
80  # for(j in 1:m){
81  #   p_0[j] <- t.test(data[1:n_ctrl, j], data[(n_ctrl+1):n, j],
        alternative = "two.sided", var.equal = TRUE)$p.value
82  # }
83
84  # True FDP
85  V <- sum(p_mat[1, (f+1):m] <= cutoff)
86  R <- sum(p_mat[1, 1:m] <= cutoff)
87  FDP <- V/R
88  FDP[is.nan(FDP)] <- 0
89
```

```
  SAM <- confSAM(p = p_mat[1, ], PM = p_mat, cutoff = cutoff, alpha = alpha
      , method = "simple")
  FDP_SAM <- SAM[3]/SAM[1]

  return(c(FDP, FDP_SAM))
}

# Set basic simulation values
seed <- 425
r <- 5000
```

Simulate different dependence structures between cases and controls.

```{r}
set.seed(seed)
sd_vec <- sqrt(seq(0, 2, 0.4))
sim_list_1 <- vector("list", length(sd_vec))
FDP_err_rate <- vector(mode = "numeric", length = length(sd_vec))

# Simulate with different correlation for two sample t-statistic
for(i in 1:length(sd_vec)){
  set.seed(seed)
  sim_list_1[[i]] <- replicate(r, simulate(n = 20, ctrl_prop = 0.5, m =
      500, sd_controls = 1, sd_cases = 1, sd_Z = sd_vec[i], f_prop = 0.1, f_
      add = 2, B = 100, cutoff = 0.01, alpha = 0.05))
}

for(i in 1:length(sd_vec)){
  greater <- sum(na.omit(sim_list_1[[i]][c(TRUE, FALSE)] > sim_list_1[[i]][
      c(FALSE, TRUE)]))
  total <- length(na.omit(sim_list_1[[i]][c(TRUE, FALSE)] > sim_list_1[[i
      ]][c(FALSE, TRUE)]))
  FDP_err_rate[i] <- greater/total
}

# Estimate for P(FDP > Upper_bound)
FDP_err_rate
1.96*sqrt(((FDP_err_rate*(1 - FDP_err_rate))/r))
```

A.6 SAM: Heteroscedasticity

```
```{r}
library(MASS)
library(confSAM)
```

Case control simulation variable variance structure
```{r}
simulate <- function(n, ctrl_prop, m, sd_controls, sd_cases, f_prop, f_add, B, cutoff, alpha){

  # Define the number of false hypotheses
  f <- f_prop*m

  # Define groupsizes
  n_ctrl <- n * ctrl_prop
  n_case <- n - n_ctrl

  # Simulate group indicator for all entries (0 = control, 1 = case)
  indicator <- c(rep(0, n_ctrl), rep(1, n_case))

  # Simulate data for cases and controls
  controls <- mvrnorm(n = n_ctrl, mu = rep(0, m), Sigma = diag(m) * ((sd_controls)^2), empirical = FALSE)
  cases <- mvrnorm(n = n_case, mu = rep(0, m), Sigma = diag(m) * ((sd_cases)^2), empirical = FALSE)

  # Merge cases and controls
  data <- rbind(controls, cases)

  # Create f 'false' hypotheses by adding f_add to cases in certain columns
  # mean shift all cases of only false hypotheses
  data[(n_ctrl+1):n, 1:f] <- data[(n_ctrl+1):n, 1:f] + (f_add * sqrt((sd_cases)^2 + (sd_controls)^2))

  # Create indicator for each permutation
  y <- t(replicate(B, sample(indicator, size = n, replace = FALSE)))
  y[1,] <- indicator  #add identity permutation

  # Permutation p-values
  p_mat <- matrix(nrow = B, ncol = m)
  for(b in 1:B){
    for(j in 1:m){
      x1 <- data[y[b, ] == 0, j]
      x2 <- data[y[b, ] == 1, j]
      n1 <- n_ctrl
      n2 <- n_case
      m1 <- mean(x1)
      m2 <- mean(x2)
```
```

```
45
46        va1 <- ((n1 - 1) * var(x1))
47        va2 <- ((n2 - 1) * var(x2))
48        sp <- sqrt((va1 + va2)/(n1 + n2 - 2))
49
50        teststat <- abs((m1 - m2)/(sp * sqrt((1/n1) + (1/n2))))
51        p_mat[b,j] <- 2 * pt(teststat, n1 + n2 - 2, lower = FALSE)
52      }
53    }
54
55    # Permutation p-values
56    # p_mat <- matrix(nrow = B, ncol = m)
57    # for(b in 1:B){
58    #   for(j in 1:m){
59    #     p_mat[b,j] <- t.test(data[y[b, ] == 1, j], data[y[b, ] == 0, j],
          alternative = "two.sided" )$p.value
60    #   }
61    # }
62
63    # Sample pvalues using permutation t-test
64    # p_0 <- vector(length = 0)
65    # for(j in 1:m){
66    #   p_0[j] <- (sum(p_mat[1, j] >= p_mat[2:B, j])+1)/B
67    # }
68
69    # Sample pvalues using t-test
70    # p_0 <- vector(length = m)
71    # for(j in 1:m){
72    #   p_0[j] <- t.test(data[1:n_ctrl, j], data[(n_ctrl+1):n, j],
          alternative = "two.sided", var.equal = TRUE)$p.value
73    # }
74
75    # True FDP
76    V <- sum(p_mat[1, (f+1):m] <= cutoff)
77    R <- sum(p_mat[1, 1:m] <= cutoff)
78    FDP <- V/R
79    FDP[is.nan(FDP)] <- 0
80
81    SAM <- confSAM(p = p_mat[1, ], PM = p_mat, cutoff = cutoff, alpha = alpha
          , method = "simple")
82    FDP_SAM <- SAM[3]/SAM[1]
83
84    return(c(FDP, FDP_SAM))
85  }
86
87  # Set basic simulation values
88  seed <- 425
89  r <- 5000
90  ```
91
```

````
Simulate different variance structures between cases and controls.
```{r}
set.seed(seed)
sd_vec <- sqrt(seq(1, 5, 0.8))
sim_list_1 <- vector("list", length(sd_vec))
FDP_err_rate <- vector(mode = "numeric", length = length(sd_vec))

# Simulate with different variance for two sample t-statistic
for(i in 1:length(sd_vec)){
  set.seed(seed)
  sim_list_1[[i]] <- replicate(r, simulate(n = 20, ctrl_prop = 0.5, m =
      500, sd_controls = sd_vec[i], sd_cases = 1, f_prop = 0.1, f_add = 2, B
       = 100, cutoff = 0.01, alpha = 0.05))
}

for(i in 1:length(sd_vec)){
  greater <- sum(na.omit(sim_list_1[[i]][c(TRUE, FALSE)] > sim_list_1[[i]][
      c(FALSE, TRUE)]))
  total <- length(na.omit(sim_list_1[[i]][c(TRUE, FALSE)] > sim_list_1[[i
      ]][c(FALSE, TRUE)]))
  FDP_err_rate[i] <- greater/total
}

# Estimate for P(FDP > Upper_bound)
FDP_err_rate
1.96*sqrt(((FDP_err_rate*(1 - FDP_err_rate))/r))
```
````

A.7 Data Illustration

```
Load packages
```{r}
library(confSAM)
library(cancerdata)
library(multtest)
library(reshape2)
library(ggplot2)
library(scales)
library(MASS)
```

Golub - read in the data
```{r}
data(golub)
X <- t(golub)
Y <- golub.cl
```

Golub - IDA
```{r}
# #Check correlations within groups compare
# cors_ctrl <- (cor(X[1:27, ]))
# cors_ctrl_up <- cors_ctrl[upper.tri(cors_ctrl, diag = FALSE)]
# cors_ctrl_vec <- as.vector(cors_ctrl_up)
#
# ncol(golub)
#
# cors_case <- (cor(X[1:38, ]))
# cors_case_up <- cors_case[upper.tri(cors_case, diag = FALSE)]
# cors_case_vec <- as.vector(cors_case_up)
#
# # Plot correlation density of each group
# df_cor <- data.frame(values = c(cors_ctrl_vec,
#                                 cors_case_vec),
#                      Groups = c(rep("ALL", length(cors_ctrl_vec)),
#                                 rep("AML", length(cors_case_vec))))
#
# ggplot(df_cor, aes(x = values, fill = Groups)) +
#   geom_histogram(position = "identity", alpha = 0.5, bins = 100) +
#   scale_x_continuous(breaks = seq(-1, 1, 0.5)) +
#   scale_y_continuous(breaks = seq(0, 160000, 40000)) +
#   labs(x = "Correlation (\u03c1)", y = "Count") +
#   theme_minimal(base_size = 13) +
#   scale_fill_manual(values = c("#21918c", "#3b528b"))
#
# # mean and sd of cor distributions
# mean(cors_ctrl)
# mean(cors_case)
```
```

```
# sd(cors_ctrl)
# sd(cors_case)

#Check heteroscedasticity within groups and compare
sds_ctrl <- apply(X[1:27, ], 2, sd)
sds_case <- apply(X[1:38, ], 2, sd)

# Plot sd density of each group
# df_sd <- data.frame(values = c(sds_ctrl,
#                                sds_case),
#                     Groups = c(rep("ALL", length(sds_ctrl)),
#                                rep("AML", length(sds_case))))
#
# ggplot(df_sd, aes(x = values, fill = Groups)) +
#   geom_histogram(position = "identity", alpha = 0.5, bins = 50) +
#   scale_x_continuous(breaks = seq(0, 2, 0.5)) +
#   scale_y_continuous(breaks = seq(0, 400, 100)) +
#   labs(x = "Standard deviation (\U03c3)", y = "Count") +
#   theme_minimal(base_size = 13) +
#   scale_fill_manual(values = c("#21918c", "#3b528b"))

mean(sds_ctrl)
mean(sds_cases)

sd(sds_ctrl)
sd(sds_cases)

df_sd_scat <- data.frame(ALL = sds_ctrl,
                         AML = sds_case)

ggplot(df_sd_scat, aes(x = ALL, y = AML)) +
    geom_point(pch = 1, color = "#3b528b") +
    geom_abline(intercept = 0, slope = 1, color = "#EF4444", linetype = "
        dashed", size = 1.05) +
    scale_x_continuous(breaks = seq(0, 2.5, 0.5), expand = c(0.01, 0.01)) +
    scale_y_continuous(breaks = seq(0, 2.5, 0.5), expand = c(0.01, 0.01)) +
    labs(x = expression(sigma[ALL]), y = expression(sigma[AML])) +
    theme_minimal(base_size = 13)
```

```
Golub - maxT single-step
```{r}
B <- 305100

# Perform MaxT and Holm
set.seed(425)
resT <- mt.maxT(X = t(X), classlabel = Y, test = "t.equalvar", side = "abs"
    , B = B)
rawp <- resT$rawp[order(resT$index)]
holm <- p.adjust(rawp, method = "holm")
```
```

```
97
98 sum(rawp < 0.05)
99 sum(sort(holm) < 0.05)
100 sum(resT$adjp < 0.05)
101
102 # Perform MaxT and Holm for Welch
103 set.seed(425)
104 resT_w <- mt.maxT(X = t(X), classlabel = Y, test = "t", side = "abs", B = B
      )
105 rawp_w <- resT_w$rawp[order(resT_w$index)]
106 holm_w <- p.adjust(rawp_w, method = "holm")
107
108 sum(rawp_w < 0.05)
109 sum(sort(holm_w) < 0.05)
110 sum(resT_w$adjp < 0.05)
111 ```
112
113 Golub - SAM
114 ```{r}
115 data_ex_SAM <- function(X, Y, B, cutoff, alpha){
116
117   # Create indicator for each permutation
118   m <- ncol(X)
119   n <- nrow(X)
120   n_ctrl <- sum(Y == 0)
121   n_case <- sum(Y == 1)
122
123   y <- t(replicate(B, sample(Y, size = n, replace = FALSE)))
124   y[1,] <- Y  #add identity permutation
125
126   # Permutation p-values
127   p_mat <- matrix(nrow = B, ncol = m)
128   for(b in 1:B){
129     for(j in 1:m){
130       x1 <- X[y[b, ] == 0, j]
131       x2 <- X[y[b, ] == 1, j]
132       n1 <- n_ctrl
133       n2 <- n_case
134       m1 <- mean(x1)
135       m2 <- mean(x2)
136
137       va1 <- ((n1 - 1) * var(x1))
138       va2 <- ((n2 - 1) * var(x2))
139       sp <- sqrt((va1 + va2)/(n1 + n2 - 2))
140
141       teststat <- abs((m1 - m2)/(sp * sqrt((1/n1) + (1/n2))))
142       p_mat[b,j] <- 2 * pt(teststat, n1 + n2 - 2, lower = FALSE)
143     }
144   }
145
```

```
  # Perform SAM
  SAM <- confSAM(p = p_mat[1, ], PM = p_mat, cutoff = cutoff, alpha = alpha
      , method = "simple")

return(SAM)
}

seed <- 425
cutoff <- c(0.05/ncol(X), seq(0.00005, 0.001, 0.00005))
SAM_results <- matrix(nrow = length(cutoff), ncol = 3)

for(i in 1:length(cutoff)){
set.seed(seed)
SAM_results[i, ] <- data_ex_SAM(X = X, Y = Y, B = 100, cutoff = cutoff[i],
    alpha = 0.05)
}

# SAM metrics
FDP_SAM <- SAM_results[, 2]/SAM_results[, 1]
FDP_SAM_upper <- SAM_results[, 3]/SAM_results[, 1]
R <- SAM_results[, 1]
FDP_SAM
FDP_SAM_upper
R

df_SAM <- data.frame(FDP = FDP_SAM,
                     FDP_up = FDP_SAM_upper,
                     c = cutoff)

ggplot(df_SAM, aes(x = c, y = FDP)) +
    geom_errorbar(aes(ymin = 0, ymax = FDP_up), width = 0.00002) +
    annotate("text", x = cutoff, y = FDP_SAM_upper + 0.002, label = R) +
    geom_point(color = "#3b528b", size = 2) +
    scale_x_continuous(labels = scales::scientific) +
    # scale_y_continuous(breaks = seq(0, 0.015, 0.045)) +
    labs(x = "c", y = expression(bar("FDP"))) +
    theme_minimal(base_size = 13)

```

```
Golub - Sequential maxT
```{r}
B <- 100
n <- length(Y)
n_ctrl <- sum(Y == 0)
n_case <- sum(Y == 1)
m <- ncol(X)
alpha <- 0.05
# Create indicator for each permutation
y <- t(replicate(B, sample(Y, size = n, replace = FALSE)))
```
```

```
y[1, ] <- Y  #add identity permutation


# Permutation matrix of test statistics
t_mat <- matrix(nrow = B, ncol = m)
for(b in 1:B){
  for(j in 1:m){
    x1 <- X[y[b, ] == 0, j]
    x2 <- X[y[b, ] == 1, j]
    n1 <- n_ctrl
    n2 <- n_case
    m1 <- mean(x1)
    m2 <- mean(x2)

    va1 <- ((n1 - 1) * var(x1))
    va2 <- ((n2 - 1) * var(x2))
    sp <- sqrt((va1 + va2)/(n1 + n2 - 2))

    t_mat[b,j] <- abs((m1 - m2)/(sp * sqrt((1/n1) + (1/n2))))
  }
}

t_mat[1,1]

# Initialize rejection vectors
R <- 1
R_new <- vector(length = 0)

# Start sequential rejection procedure
while(setequal(R, R_new) == FALSE){

  # Update rejection vector
  R <- R_new

  # Get rejection quantile
  Tmax <- apply(t_mat, 1, max, na.rm = TRUE)
  q <- sort(Tmax)[(1 - alpha) * B]

  # Select set of hypotheses to consider
  hypotheses <- 1:m
  if(length(R) > 0){
    hypotheses <- hypotheses[-R]
  }

  # Reject hypotheses based on quantile
  for(j in hypotheses){
    if(t_mat[1, j] > q){
      t_mat[, j] <- NA
      R_new <- c(R_new, j)
    }
```

```
  }
}

length(R)


# Welch version
# Create indicator for each permutation
y <- t(replicate(B, sample(Y, size = n, replace = FALSE)))
y[1, ] <- Y  #add identity permutation


# Permutation matrix of test statistics (Welch)
t_mat <- matrix(nrow = B, ncol = m)
for(b in 1:B){
  for(j in 1:m){
    x1 <- X[y[b, ] == 0, j]
    x2 <- X[y[b, ] == 1, j]
    n1 <- n_ctrl
    n2 <- n_case
    m1 <- mean(x1)
    m2 <- mean(x2)

    sd1 <- sd(x1)
    sd2 <- sd(x2)
    sdx1 <- sd1/sqrt(n1)
    sdx2 <- sd2/sqrt(n2)

    t_mat[b,j] <- abs((m1 - m2)/(sqrt((sdx1^2) + (sdx2^2))))
  }
}

t_mat[1,1]

# Initialize rejection vectors
R_w <- 1
R_new <- vector(length = 0)

# Start sequential rejection procedure
while(setequal(R_w, R_new) == FALSE){

  # Update rejection vector
  R_w <- R_new

  # Get rejection quantile
  Tmax <- apply(t_mat, 1, max, na.rm = TRUE)
  q <- sort(Tmax)[(1 - alpha) * B]

  # Select set of hypotheses to consider
  hypotheses <- 1:m
```

```
  if(length(R) > 0){
    hypotheses <- hypotheses[-R_w]
  }

  # Reject hypotheses based on quantile
  for(j in hypotheses){
    if(t_mat[1, j] > q){
      t_mat[, j] <- NA
      R_new <- c(R_new, j)
    }
  }
}

length(R)
```

Introduction - Plot of FWER as function of m0 under independence

```{r}
m0 <- seq(0, 100, 5)
alpha <- 0.05
fwer <- 1 - (1 - alpha)^(m0)
fwer_df <- data.frame(fwer = fwer,
                      m0 = m0)

ggplot(data = fwer_df, aes(x = m0, y = fwer)) +
  geom_line(aes(y = fwer), color = "blue") +
  geom_point(aes(y = fwer), size = 2) +
  geom_hline(yintercept = alpha, linetype = "dashed", color = "red") +
  labs(x = expression('m'[0]), y = "F␣W␣E␣R") +
  theme_minimal(base_size = 13)
```
