ETH
Swiss Federal Institute of Technology Zurich

Seminar for Statistics

Department of Mathematics

Semester Paper Fall 2018

Tianqi Wang

A Comparison on Several Approaches to Multiple Testing Problem

Submission Date: December 19th 20018

Adviser: Dr. Martin Maechler

Abstract

Multiple testing problems widely appear in many fields, such as genomics, marketing, and astronomy. For one instance, the identification of co-expressed genes in high-throughput gene expression experiments would require thousands or even millions hypotheses to be tested simultaneously(S.Dudoit and der Laan).

When multiple comparisons are made simultaneously, the probability of false discovery increases and it would lead to wrong conclusions. Therefore, many adjustment procedures are introduced to correct for this multiple testing problem. Some of the methods control family-wise error rate(FWER) and others control false discovery rates(FDR). In this paper, we introduce these two types of procedures and compare them in terms of their ability to control the false discoveries and their statistical power.

We also carry out a simulation study to support the conclusion. The simulation study is conducted in dependent as well as independent cases. This is closer to the real world case, as in practice microarray studies typically involve dependent test statistics(S.Dudoit and der Laan).

Our result shows that in general FWER-controlling methods are more conservative, in that they lead to less false rejection but less statistical power.

Contents

Chapter 1

Background

In statistics, a hypothesis is an assumption about the property of the unknown data generating distribution. Hypothesis testing refers to the procedure that one uses the observed data to make decisions as to whether reject the statistical hypothesis or not.

The result of a hypothesis test is often reported in terms of p-value, which is defined as the probability of obtaining a result that is equal to, or more extreme than, the observed value under null hypothesis. The null hypothesis is rejected when the p-value is smaller than some fixed level α, namely the significance level. Type 1 error is induced when we reject the null hypothesis when it is true, and type 2 error occurs when we fail to reject the null hypothesis when it is false.

Assume that there are m null hypotheses, denoted by $H_1, H_2, ..., H_m$. By applying hypotheses testing procedures, four possible type of outcomes could be given. Summarizing each of these results would yield the following table:

	Null hypothesis is true	Alternative hypothesis is true	total
Test declared significant	V	S	R
Test declared non-significant	U	T	m - R
Total	m_0	$m - m_0$	m

Table 1.1: possible outcomes for multiple hypotheses testing

In hypotheses testing, we would like to minimize both the type 1 and type 2 error. Typically, this is achieved by controlling type 1 error at fixed level α and search for the rejection region that minimize type 2 error. In multiple hypotheses testing, however, each test has a type 1 and type 2 error and it becomes unclear how to measure the overall error rate.

Under this context, two commonly used error rates are introduced, namely family-wise error rate(FWER) and false discovery rate(FDR).

Family-wise error rate is defined as the probability of rejecting at least one true null hypothesis, and is computed as:

$$FWER = P(V \geq 1)$$

False discovery rate is the expected proportion of false discoveries among all discoveries, and is thus computed as:

$$FDR = E(V/max(R, 1))$$

In general, for a given multiple testing procedure, false discovery rate is smaller or equal to family-wise error rate. However, when all null hypotheses are true, these two are equivalent. This can be shown as follows:

$$\begin{aligned} FDR &= E(\frac{V}{max(R, 1)}) \\ &\leq E(\frac{V}{max(V, 1)}) \\ &= P(V \geq 1) \\ &= FWER \end{aligned}$$

Therefore, procedures controlling FWER are said to be more conservative than FDR controlling procedures in that leads to fewer rejected hypothese, which generally decreases false rejections at the cost of a reduced statistical power.

Benjamini and Hochberg (1995)

Chapter 2

Methods Controlling Family-wise Error Rate

2.1 single-step procedures

For a single-step procedure, each null hypothesis is tested with a rejection region that is independent of the results from the tests of other hypotheses.The cost of such a conservative approach is an increased probability in committing type 2 error, which means a weaker statistical power.

2.1.1 Bonferroni Correction

Definition 1. Bonferroni Procedure *By Bonferroni Correction, the adjusted significance level follows as* $\alpha' = \frac{\alpha}{m}$*. Or equivalently, the adjusted p-value* $p' = min\{mp, 1\}$*.*

Bonferroni Correction controls family-wise error rate at level smaller or equal to α. The proof could follow from Boole's inequality:

$$FWER = P\{\bigcup_{i=1}^{m_0} p_i \leq \frac{\alpha}{m}\} \leq \sum_{i=1}^{m_0} P(p_i \leq \frac{\alpha}{m}) = m_0\frac{\alpha}{m} \leq m\frac{\alpha}{m} = \alpha$$

When the test statistics are positively dependent, the upper bound on family-wise error rate still holds, because Bool's equality does not make assumptions on independence. However, the procedure is considered very conservative in such a case, since the information from the dependency structure is totally ignored.

2.2 stepwise procedures

To improve the statistical power while preserving the control for the family error rate at α, the notion of stepwise procedures is introduced. Stepwise procedure allows one to make decision to reject the null hypotheses based on the result others.

2.2.1 Holm Step-down Procedure

Definition 2. Holm Step-down Procedure(Holm (1979))
step1: sort p-values in ascending order
step2: starting from $k = 1, ..., m$, $kh = min\{k : p_{(k)} > \frac{\alpha}{m+1-k}\}$
step3: if no such k, reject all null hypotheses. If $k = 1$*, don't reject any. Otherwise, reject the first k - 1 null hypotheses.*

The intuition of Holm procedure can be motivated as follows. The procedure sorts the p-values ascendingly as $P_{(1)}, P_{(2)}, ..., P_{(m)}$, compares them to the nominal α levels at $\frac{\alpha}{m}$ $\frac{\alpha}{m-1}$ $...\frac{\alpha}{1}$, and keeps the sequential rejection procedure as long as p-value does not exceed the given level. Compared with Bonferroni Correction, this method relieves the criteria for null hypothesis rejection and thus obtains more statistical power. However, as suggested by Holm[7], this increase in power is not very large.

Meanwhile, Holm procedure ensures a stronger level α FWER control than Bonferroni Correction. This could be seen as following:

Let $H_1, ..., H_m$ be a family of null hypotheses sorted corresponding to the ascending order of p-values. Let j be the first false rejected null hypothesis according to the Holm Procedure, and then $H_1, ..., H_{j-1}$ would be the set of rejected false null hypotheses. It follows that $j - 1 \leq m - m_0$. Therefore, $P_{(j)} \leq \frac{\alpha}{m-j+1} \leq \frac{\alpha}{m_0}$. Then, the result is again followed from Bool's inequality.

Alternatively, we could also see Holm Procedure as a closing Bonferroni Correction. The notion of closed testing procedure is proposed in R. Marcus (1976):

Suppose there are m hypotheses $H_1, ..., H_m$ *to be tested and the overall type 1 error rate is* α*. The closed testing principle allows the rejection of any one of these elementary hypotheses, say* H_i*, if all possible intersection hypotheses involving* H_i *can be rejected by using valid local level* α *tests.*

The reason that Holm is a closed Bonferroni can be seen as following:

Assume that $H_1, ..., H_n$ corresponds to the ascending order of p-values. Consider an index set I, for which $p_{(j)}$ is the smallest p-value in with index in I, and the mth smallest value in total. Let φ_I denote a α level test for H_I and rejects if $\varphi_i = 1$. Let $I_{(j)+}$ denote the index set corresponding to the $n-j+1$ th largest p-values. It can be proved that $\varphi_{I^+_{(j)}} = 1$ implies that $\varphi_I = 1$.

Then, we could see that sequentially,

$$\begin{aligned} H_j isrejected &\Leftrightarrow \varphi_{I^+_{(1)}} = 1, ..., \varphi_{I^+_{(j)}} = 1 \\ &\Leftrightarrow \varphi_{I_{(1)}} = 1, ..., \varphi_{I_{(j)}} = 1 \\ &\Leftrightarrow p_{(1)} \leq \frac{\alpha}{n}, ..., p_{(j)} \leq \frac{\alpha}{n-j+q} \end{aligned}$$

A nice feature of closure principle is that is controls FWER strongly. This reinforces the statement above.

2.2.2 Hochberg Step-up procedure

Definition 3. Hochberg Step-up Procedure(Hochberg (1995))
step1: sort p-values in ascending order
step2: starting from $k = m, ..., 1$, $k = mmax\{k : p_{(k)} \leq \frac{\alpha}{m+1-k}\}$
step3: if such k exists, reject all null hypotheses $H_1, ..., H_k$*. Otherwise, don't reject any.*

Although Hochberg procedure relies on the same p-value cutoff as Holm, it is typically more powerful. The reason is that the order at which null hypothese are tested matters. Hochberg Procedure first considers the least significant null hypothses and, as soon as one hypothesis is rejected, reject the remaining more significant ones. Thus, it gives each null hypothesis more chances at rejection and potentially leads to a greater number of rejected hypotheses. An extreme case would be that when every p-value is α. In such situation, Hochberg procedure rejects every null hypothesis, while Holm procedure rejects none.

Chapter 3

Methods Controlling False Discovery Rate

3.1 Non-adaptive Procedures

3.1.1 Benjamini and Hochberg Linear Step-up procedure

Definition 4. BH Procedure(Benjamini and Hochberg)
step1: sort p-values in ascending order
step2: starting from $k = 1, ..., m$, $k = max\{k : P_{(k)} \leq \frac{k}{m}\alpha\}$
step3: if such k exists, reject $H_{(1)}...H_{(k)}$. *Otherwise, don't reject any.*

For independent test statistics, BH controls FDR at level α, or more precisely, at $\alpha m_0/m$[1]. From an empirical view, we could also see BH procedure as a method that aims to equate the empirical FDR to the significance level α. This can be shown as follows:

Take a fixed t, and reject H_i if and only if $P_{(i)} \leq t$. Define the empirical CDF of p-values as:

$$\hat{F_m}(t) = \frac{\#\{i : p_{(i)} \leq t\}}{m}$$

Using the definition of BH procedure above, we could write the critical p-value as

$$P_{BH} = max\{p(i) : p(i) \leq \alpha\frac{i}{m}\} = max\{p(i) : p(i) \leq \alpha\hat{F_m}p(i)\} = max\{t : t \leq \alpha\hat{F_n}t\}$$

Equivalently, we could write

$$T_{BH} = max\{t : \frac{t}{max(\hat{F_n}(t), 1/n)} \leq \alpha\}$$

Recall that

$$FDR(t) = E[FDP(t)] = E[\frac{V(t)}{max(R(t), 1)}]$$

$R(t)$, which is the number of rejections, is known. Therefore, the only problem is that $V(t)$, the number of false rejections remains unknown. We could take advantage of the fact that $E[V(t)] = m_0 t$, but still m_0 is unknown. A conservative way to deal with it would be to use m as an estimation of m_0. This leads to:

$$\hat{FDR}(t) = \frac{mt}{max(R(t)), 1} = \frac{t}{max(\hat{F_n}(t), 1/n)}$$

This gives us that

$$\hat{FDR}(T_{BH}) = \frac{T_{BH}}{max(F_n(\hat{T}_{BH}), 1/n)} \leq \alpha$$

Benjamini and Yekutieli extended the original paper by Benjamini and Hochberg and investigated the procedure under dependence. They show that under dependence, BH procedure controls FDR at level $\alpha S(n)$, or more specifically,

$$\hat{FDR} \leq \alpha S(n), \text{ where } S(n) = 1 + 1/2 + 1/3 + ... + 1/n$$

If BH procedure is conducted at level $\alpha' = \frac{\alpha}{S(n)}$, it controls FDR at level α. This concludes the motivation for BY method.

Definition 5. BY Procedure(Benjamini and Yekutieli (2001))
step1: sort p-values in ascending order
step2: starting from $k = 1, ..., m$, $k = max\{k : P_{(k)} \leq \frac{k}{m}\frac{\alpha}{\sum_{i=1}^{m}\frac{1}{i}}\}$
step3: if such k exists, reject $H_{(1)}...H_{(k)}$. *Otherwise, don't reject any of the hypothesis.*

3.2 Adaptive Procedures

m_0, which is the number of true null hypotheses, can be very important for improving on the performance of FDR controlling procedure. As shown above, BH procedure controls FDR at $\frac{m_0}{m}\alpha$. If m_0 is known, α can be adjusted by $\alpha' = \frac{m}{m_0}\alpha$, which controls FDR precisely at desired level α in the independent case. In reality, even though m_0 cannot be known, there are various ways to estimate it.

3.2.1 Benjamini & Hochberg linear step-up adaptive procedures

Definition 6. BH Linear Step-up Adaptive method(Benjamini and Hochberg (2000))
step1: Use the linear step-up procedure at level . If no hypothesis is rejected, stop and reject none. Otherwise, proceed to the second step.
step2: estimate $m_0(k) = \frac{m-k+1}{1-p_{(k)}}$
step3: starting with $k = 2...m$, *stop when* $m_0(k) > m_0(k-1)$
step4: estimate $\hat{m_0} = min\{\hat{m_0}, m\}$
step5: use linear step-up procedure with $\alpha' = \frac{m}{\hat{m_0}}\alpha$

The motivation for step 2 is as follows, as shown in the paper:

let $r(\alpha)$ be the number of rejected null hypotheses at significance level α. αm is the number of false rejections. It follows that $m_0 \approx m-(r(\alpha)-\alpha m_0)$. This gives that $m_0(\alpha) \approx \frac{m-r(\alpha)}{1-\alpha}$. Plugging in $\alpha = p_k$ gives the approximation to $\alpha = p_{(k)}$ gives the approximation as in step 2.

It is also suggested that an estimator of the above form evaluated at a single prespecified α quantiles or certain cutoff λ can be easier to study(Benjamini, Krieger, and Yekutieli (2006)). This lead to the following procedures.

Definition 7. Linear Step-up Adaptive method with Storey's estimator(Storey (2002))
step1: Let $r(\lambda) = \#\{p_{(i)} \leq \lambda\}$.
step2: estimate m_0 *by* $\hat{m_0}(k) = \frac{m-r(\lambda)}{1-\lambda}$
step3: use linear step-up procedure with $\alpha' = \frac{m}{\hat{m_0}}\alpha$

Storey motivates his method as follows. Instead of previous procedures that fix type 1 error and estimate the rejection region that minimizes type 2 error, Storey fixes the rejection region, which is λ in this case, and then estimate α. In the paper he shows that this method as well provides a conservative bias in expectation.

This result can be strange when all the p-values are smaller than or equal to λ. Specifically, in such a case, $\hat{m_0}$ would be 0 and this would adjust all p-values to exactly 0. In a follow-up paper(J.D. Storey (2004)), Storey proposed to replace $\hat{m_0}(k) = \frac{m-r(\lambda)}{1-\lambda}$ with $\hat{m_0}(k) = \frac{m-r(\lambda)+1}{1-\lambda}$, and this can resolve the problem. Plus, Storey shows that with this modification, together with the constraint that the significance threshold is in the region [0, λ], the control can be proved in the finite sample case. However,these additional constraints are not needed for the proof in the asymptotic case.

Storey also suggested that strong control of FDR can be proved for this proposed procedure, but not for the Adaptive BH procedure.

The choice of optimal λ results in a bias-variance tradeoff. As λ becomes larger, the bias of $\hat{m_0}/m$becomes smaller but the variance becomes larger.One way to pick the optimal λ is by minimzing the mean squared error of our estimation of FDR:

$$\lambda_{best} = argmin(E[FD\hat{R}_\lambda(\gamma) - FDR_{(\gamma)}{}^2])$$

We could achieve this by bootstrapping.

Definition 8. Quantile Adaptive Linear step up method(Efron, Tibshirani, Storey, and Tusher (2001))
step1: estimate m_0 *by* $\hat{m_0}(k) = \frac{m-qm}{1-p_{(qm)}}$
step2: use linear step-up procedure with $\alpha' = \frac{m}{\hat{m_0}}\alpha$

When $q = 1/2$, this method corresponds to the median adaptive linear step up method.

3.3 Two Stage Procedures

Definition 9. The two-stage linear step-up procedure(Storey (2002))
step 1: Use the linear step-up procedure at level $\alpha' = \alpha/1+\alpha$. *Let* r_1 *be the number of rejected hypotheses. If* $r_1 = 0$ *do not reject any hypothesis and stop. If* $r_1 = m$ *reject all*

m hypotheses and stop. Otherwise, continue.
step 2: Let $\hat{m_0} = m - r_1$.
step 3: Use the linear step-up procedure with $\alpha^* = \alpha' m / \hat{m_0}$.

The motivation for TST is that the value of m_0 can be estimated from the result of the one-step procedure. This method examines the distribution of p-values for an estimation of the fraction of true null hypotheses in the first stage, and then uses this information to gain more power when deciding when a p-value is low enough to be a discovery.

It is shown that TST procedure controls FDR below but close to the nominal level α, and provides higher power than the original BH procedure when the tests are correlated.

Chapter 4

Simulation

4.1 Adjusted P-value

In this paper, we compare the procedures mentioned above, where 1-3 are family-wise error rate controlling methods, 4-5 are non-adaptive false discovery rate controlling methods, 7-8 are adaptive procedures, and 9-10 are the two stage methods.

- Procedure 1. Bonferrorni Correction, denoted by "bonferroni"
- Procedure 2. Holm step-down procedure, denoted by "holm"
- Procedure 3. Hochberg step-up procedure, denoted by "hochberg"
- Procedure 4. Benjamini and Hochberg linear step-up procedure, denoted by "BH"
- Procedure 5. Benjamini and Yekutieli procedure, denoted by "BY"
- Procedure 6. Benjamini and HochBerg adaptive linear step-up procedure, denoted by "ABH"
- Procedure 7. Median adaptive linear step-up procedure, denoted by "MBH"
- Procedure 8. Adaptive linear step-up procedure with Storey's estimator, denoted by "storey"
- Procedure 9. Two stage step-up procedure, denoted by "TST"
- Procedure 10. Modified two stage step-up procedure, denoted by "MTST"

First of all, we generate 31 artifical p-values, which are equally spaced in the log scale. We can easily visualize the adjusted p value against the original p-value for each of these methods. From this plot below, we can get a sense of how the adjusted value under these methods behave.

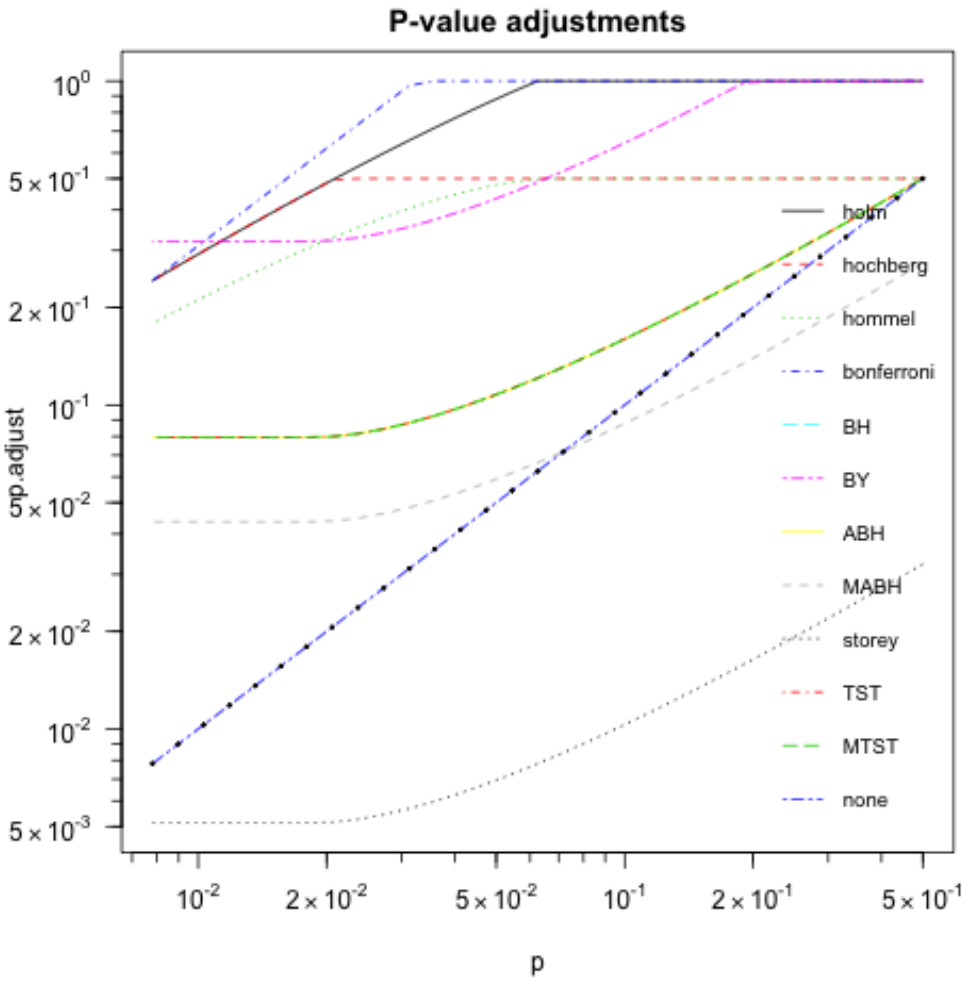

Figure 4.1: adjusted p-values vs. p-values for 30 equidistant p-values in log-scale

4.2 R packages

In order to derive the results from the above procedures, we extended the original p.adjust() method in R by including procedure 6-10. This code can be found in the appendix.

For a sanity-check, we used the results obtained from R packages multtest and mutoss as comparisons. Multtest package has an implementation of the adaptive BH procedure and two stage procedure, and Mutoss package implements the adaptive BH procedure and adaptive BH procedure with Storey's estimator. Since the procedures 7 and 10 are simple variations of the above methods, there should be no problem with them if ABH and TST are shown to be fine.

First of all we compare the ABH method. In the code below, we covered two situations: the first one goes through step 1 to 5 from the definition of ABH above, and the other case occurs when we cannot find a k such that $m_0(k) > m_0(k-1)$.

```
> library(multtest)
> library(mutoss)
```

```
> source("p_adjust.R")
> set.seed(123)
> # first case
> x <- rnorm(50, mean=c(rep(0,25), rep(3,25)))
> p <- 2*pnorm(sort(-abs(x)))
> p_ABH_multtest <- mt.rawp2adjp(p, proc='ABH', alpha=0.05)
> p_ABH_multtest <- p_ABH_multtest$adjp[,2][order(p_ABH_multtest$index)]
> p_ABH_adjust <- p.adjust(p,"ABH")
> p_ABH_mutoss <- adaptiveBH(p, 0.05, silent=TRUE)$adjPValues
> all.equal(p_ABH_multtest, p_ABH_adjust)

[1] "Mean relative difference: 0.03703704"

> all.equal(p_ABH_mutoss, p_ABH_adjust)

[1] TRUE

> # second case
> p1 <- c(0.05, 0.01, 0.003, 0.009, 0.1)
> p_ABH_multtest1 <- mt.rawp2adjp(p1, proc='ABH', alpha=0.05)
> p_ABH_multtest1 <- p_ABH_multtest1$adjp[,2][order(p_ABH_multtest1$index)]
> p_ABH_padjust1 <- p.adjust(p1, "ABH")
> p_BH_padjust1 <- p.adjust(p1,"BH")
> p_ABH_mutoss1 <- adaptiveBH(p1, 0.05, silent=TRUE)$adjPValues
> all.equal(p_ABH_padjust1, p_BH_padjust1)

[1] TRUE

> all.equal(p_ABH_padjust1, p_ABH_mutoss1)

[1] TRUE

> p_ABH_multtest1

[1] NA NA NA NA NA
```

From the results above, we could see that our result is consistent with mutoss package, but different from multtest package. In the first case, our result is a little bit off from the method from multtest package, and in the second case, multtest mistakenly gives NAs. Then I went through the source code of multtest package and find the following two lines to be potentially problematic.

```
> grab <- min(which(diff(h0.m,na.rm=TRUE)>0),na.rm=TRUE)
```

According to the BH prodecure mentioned above, we would like to find the smallest k that satisfies m0(k) > m0(k-1), starting from k=2. However, what the code here above does is to find the smallest k such that m0(k+1) > m0(k), starting from k=1. Therefore, I think that this method will find k-1 rather than k proposed by the original method.

```
> h0.ABH <- ceiling(min(h0.m[grab],mgood))
```

Another problem is that if such k does not exist, we would like to replace m0 with m, and then this method falls back to the original linear step up procedure(BH method). However, the multtest package this would produce NAs instead.

Then we compare our implementation of two-stage method with the multtest package and find out that the two results agree.

```
> p_TSBH_multtest <- mt.rawp2adjp(p, proc='TSBH', alpha=0.05)
> p_TSBH_multtest <- p_TSBH_multtest$adjp[,2][order(p_TSBH_multtest$index)]
> p_TSBH_padjust <- p.adjust(p,"TST")
> all.equal(p_TSBH_multtest, p_TSBH_padjust)

[1] TRUE
```

For the adaptive BH with Storey's estimator, our implementation and mutoss package gives identical result in the first two cases. But surprisingly in the third case, these two implementation gives different result. I looked into the source code of mutoss package and found out the reason for this difference. In our implementation, I followed the definition above and adjust p-value with $p\_adj = min(p\hat{m}_0/m, 1)$, where m_0 is derived from step 2. The mutoss package adjusts the p-value first by $p\_adj = p/m$ and multiplied this value by $min(m_0/m, 1)$. Since $m_0/m > 1$, these two implementation will yield different results. From my perspective the difference is negligible, but I'm not sure which one makes more sense.

```
> p_STS_adjust <- p.adjust(p,"storey")
> p_STS_mutoss <- adaptiveSTS(p, 0.05, silent=TRUE)$adjPValues
> all.equal(p_STS_mutoss, p_STS_adjust)

[1] TRUE

> p2 <- c(0.1, 0.2, 0.3, 0.4, 0.5)
> p_STS_adjust <- p.adjust(p2,"storey")
> p_STS_mutoss <- adaptiveSTS(p2, 0.05, silent=TRUE)$adjPValues
> all.equal(p_STS_mutoss, p_STS_adjust)

[1] TRUE

> p3 <- c(0.6, 0.7, 0.8, 0.9)
> p_STS_adjust <- p.adjust(p3,"storey")
> p_STS_mutoss <- adaptiveSTS(p3, 0.05, silent=TRUE)$adjPValues
> all.equal(p_STS_mutoss, p_STS_adjust)

[1] "Mean relative difference: 0.1111111"
```

4.3 Simulations

A simulation study is performed under a similar setting as the one in [4].

In addition to the procedures above, a "oracle" procedure is also included.

- Procedure 11. Linear step-up procedure at level $\alpha m_0/m$, denoted by "oracle"

The "oracle" procedure is not implementable in reality, since m_0 cannot be known in advance. In our case, it mainly serves as a benchmark for method comparison.

For our simulation study, we assume the null and alternative hypothesis to be:

$$H_0 : \mu = 0$$
$$H_1 : \mu \neq 0$$

The way we generate p-values is as follows. First, we generated $Z_0, Z_1, ...Z_m$ iid from $N(0,1)$. Let $Y_i = \sqrt{\gamma}Z_0 + \sqrt{1-\gamma}Z_i - \mu_i$. Then, it follows that $P_i = 1 - W(Y_i)$.

In our simulation, two configurations of μ are implemented: "all at 5" configuration assumes that μ of all false null hypotheses equals to 5, and "1234" configuration assumes them to be a repeated cycle of "1234". Besides, we're considering number of tests m at value 4, 8, 16, 32, 64, 128, 256, 512, the correlation coefficient γ at 0, 0.25, 0.5, 0.75, and the fraction of true null hypotheses r at level, 25%, 50%, 75%.

4.3.1 Independent Test Statistics

First, let us take a look at the result with "all-at-5" configuration.

	m0/m = 0.75			m0/m = 0.5			m0/m = 0.25		
	m=16	m=64	m=256	m=16	m=64	m=256	m=16	m=64	m=256
bonferroni	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
hochberg	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
holm	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
BH	0.033	0.034	0.037	0.023	0.024	0.025	0.011	0.012	0.012
BY	0.003	0.003	0.003	0.003	0.002	0.002	0.001	0.001	0.001
ABH	0.042	0.046	0.049	0.046	0.047	0.048	0.045	0.048	0.049
MABH	0.043	0.047	0.050	0.046	0.048	0.049	0.023	0.025	0.025
storey	0.043	0.047	0.050	0.047	0.048	0.049	0.046	0.050	0.050
TST	0.044	0.047	0.050	0.048	0.049	0.051	0.048	0.051	0.052
MTST	0.041	0.044	0.048	0.045	0.047	0.048	0.046	0.049	0.050
oracle	0.043	0.046	0.050	0.047	0.048	0.049	0.047	0.050	0.050

Table 4.1: Simulation study with independent testing statistics and "all-at-5" μ configuration. The numbers shown in the table are estimated values of FDR for selected values of m and r

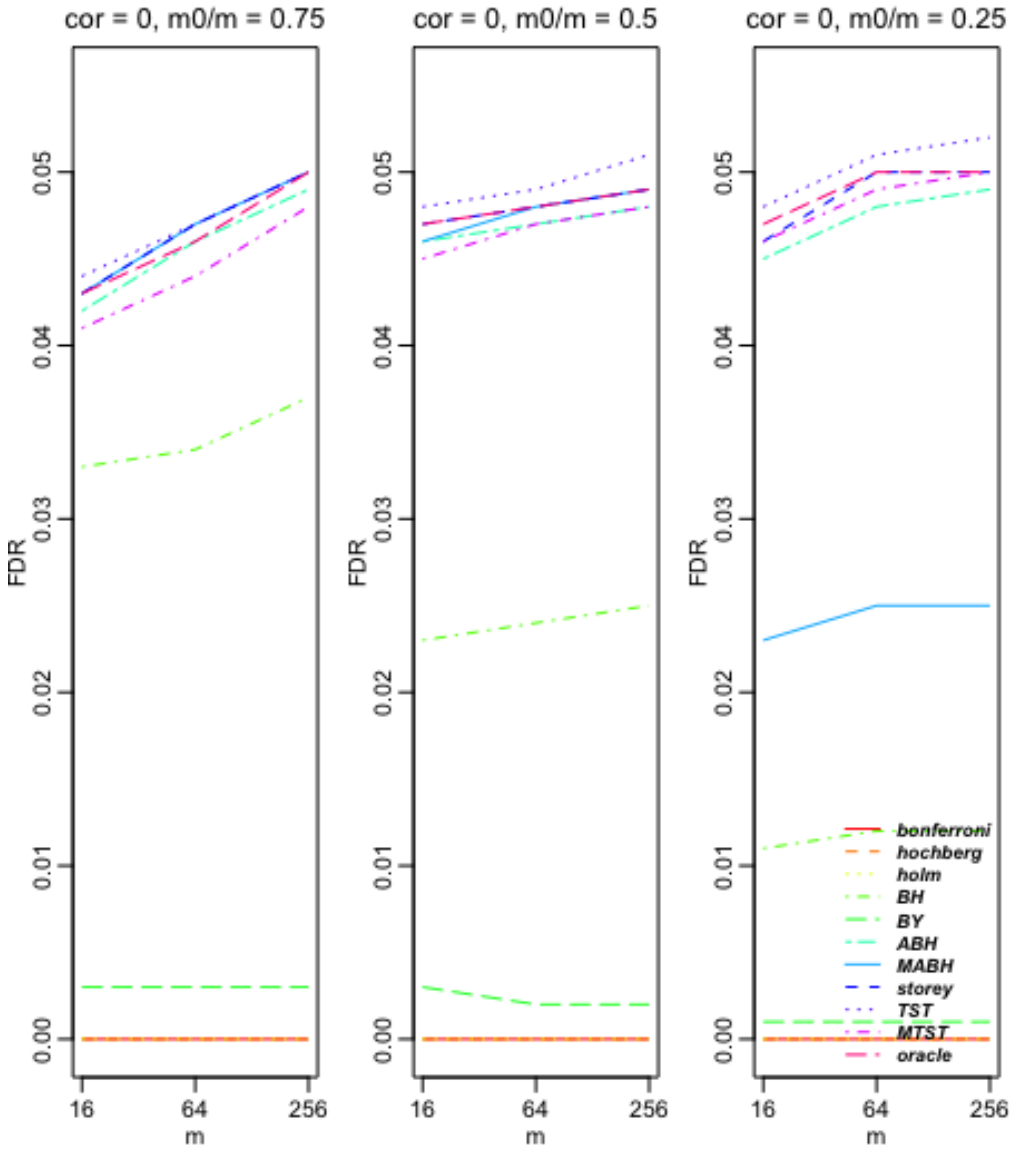

Figure 4.2: Simulation study with independent testing statistics and "all-at-5" configuration

	m0/m = 0.75			m0/m = 0.5			m0/m = 0.25		
	m=16	m=64	m=256	m=16	m=64	m=256	m=16	m=64	m=256
bonferroni	0.633	0.525	0.421	0.627	0.526	0.421	0.632	0.523	0.421
hochberg	0.646	0.536	0.428	0.659	0.551	0.439	0.690	0.565	0.451
holm	0.646	0.536	0.428	0.659	0.551	0.439	0.690	0.565	0.451
BH	0.995	0.993	0.994	0.997	0.997	0.997	0.998	0.998	0.998
BY	0.961	0.957	0.955	0.976	0.972	0.971	0.982	0.980	0.978
ABH	0.996	0.994	0.996	0.999	0.999	0.999	1.000	1.000	1.000
MABH	0.996	0.994	0.996	0.999	0.999	0.999	0.999	0.999	0.999
storey	0.996	0.994	0.996	0.999	0.999	0.999	1.000	1.000	1.000
TST	0.996	0.995	0.996	0.999	0.999	0.999	1.000	1.000	1.000
MTST	0.996	0.994	0.995	0.999	0.999	0.999	1.000	1.000	1.000
oracle	0.996	0.994	0.996	0.999	0.999	0.999	1.000	1.000	1.000

Table 4.2: Simulation study with independent testing statistics with ”all-at-5” configuration. The numbers shown in the table are estimated values of power for selected values of m and r

Below are the results of FDR and Power for "1234" configuration.

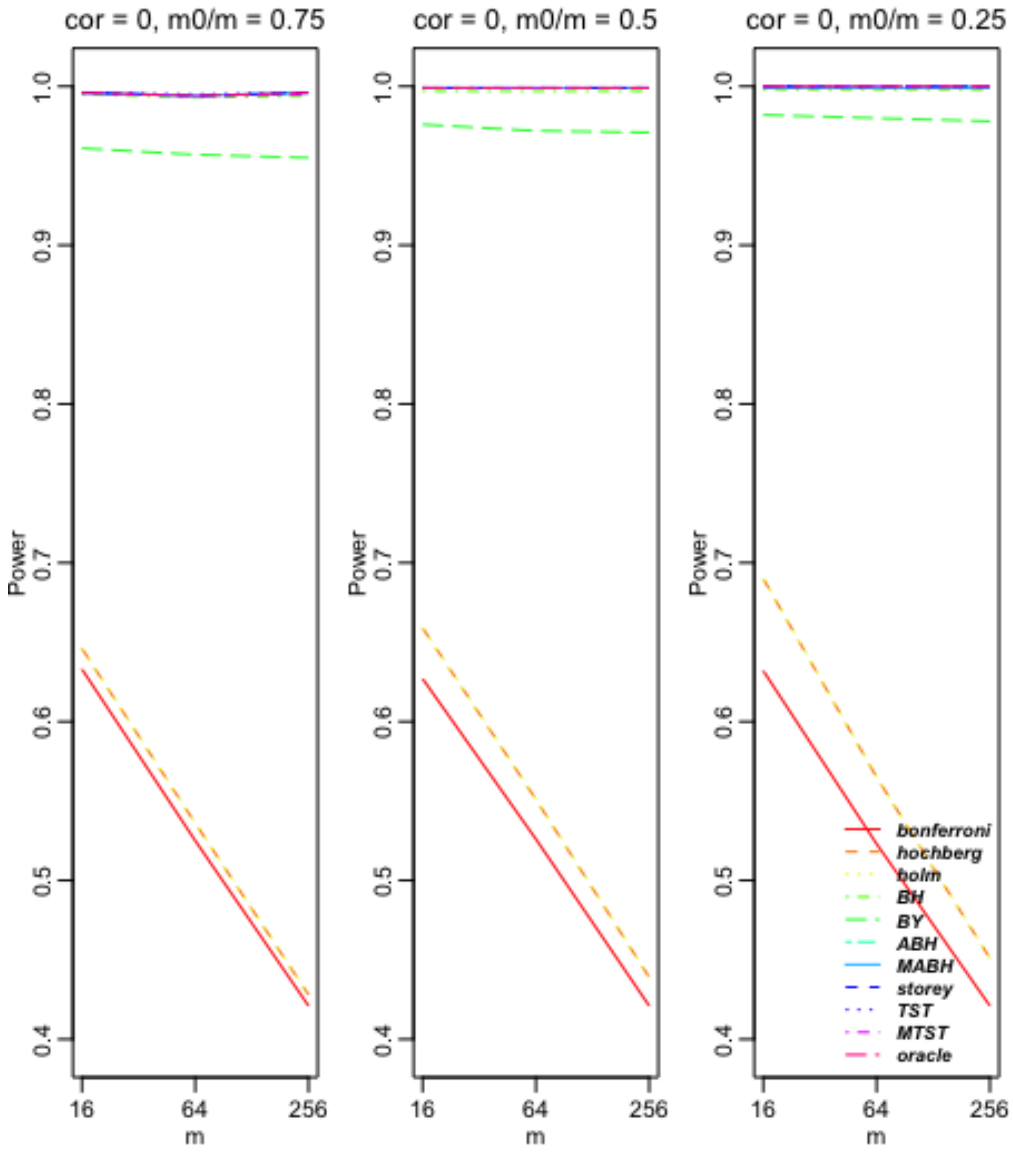

Figure 4.3: Simulation study with independent testing statistics with "all-at-5" configuration

From these tables, we could see that family-wise error rate controlling procedures are significantly more conservative, in that they reject fewer hypotheses and none false positive. As a tradeoff, these methods have weak statistical powers. We could also see that BY and BH methods are very conservative, but compared with FWER controlling procedures, with these methods we could attain relatively higher statistical power.

With a decreasing fraction of true null hypotheses, we observe that the median adaptive BH method starts performing noticeably more conservatively. The reason might be that the median fails to be a fair estimation as the fraction changes. Changing the cutoff quantiles might improve the performance, but without a prior knowledge of m_0 it might

be hard to do so. Adaptive Linear Step-up procedures and the two stage procedures perform relatively consistent with the decreasing fraction of true null hypotheses.

We could also observe that with an increasing number of tests, FDR typically gets closer to the significance level α. Power tends to decrease for FWER controlling methods significantly because they are more conservative with an increased m.

	m0/m = 0.75			m0/m = 0.5			m0/m = 0.25		
	m=16	m=64	m=256	m=16	m=64	m=256	m=16	m=64	m=256
bonferroni	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
hochberg	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
holm	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
BH	0.029	0.030	0.038	0.021	0.022	0.025	0.012	0.013	0.013
BY	0.002	0.002	0.003	0.003	0.002	0.002	0.001	0.001	0.001
ABH	0.032	0.034	0.042	0.029	0.030	0.032	0.019	0.019	0.020
MABH	0.036	0.039	0.046	0.035	0.037	0.039	0.021	0.023	0.023
storey	0.036	0.039	0.046	0.036	0.038	0.041	0.028	0.030	0.031
TST	0.032	0.035	0.042	0.029	0.032	0.034	0.019	0.020	0.021
MTST	0.032	0.033	0.041	0.029	0.030	0.032	0.018	0.019	0.020
oracle	0.039	0.041	0.050	0.045	0.046	0.049	0.046	0.050	0.050

Table 4.3: Simulation study with independent testing statistics with ”1234” configuration. Estimated values of FDR for selected values of m and r.

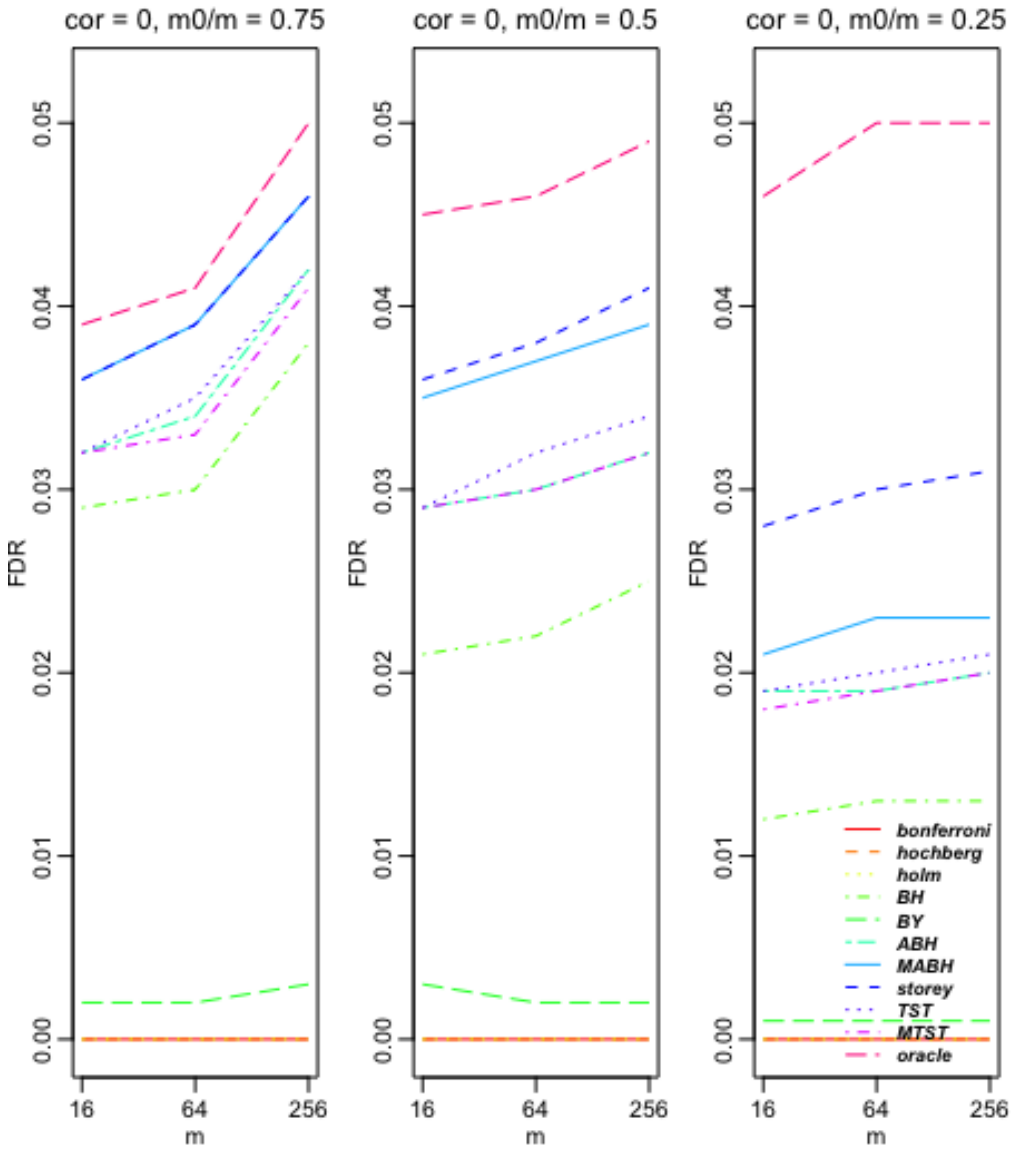

Figure 4.4: Simulation study with independent testing statistics with ”1234” configuration

	m0/m = 0.75			m0/m = 0.5			m0/m = 0.25		
	m=16	m=64	m=256	m=16	m=64	m=256	m=16	m=64	m=256
bonferroni	0.074	0.054	0.032	0.070	0.052	0.032	0.074	0.051	0.032
hochberg	0.074	0.054	0.032	0.071	0.052	0.033	0.075	0.052	0.032
holm	0.074	0.054	0.032	0.071	0.052	0.033	0.075	0.052	0.032
BH	0.442	0.436	0.437	0.507	0.507	0.505	0.545	0.545	0.547
BY	0.230	0.238	0.229	0.288	0.288	0.276	0.322	0.318	0.307
ABH	0.456	0.447	0.448	0.540	0.535	0.533	0.600	0.601	0.602
MABH	0.467	0.457	0.459	0.556	0.557	0.554	0.613	0.616	0.621
storey	0.467	0.457	0.459	0.562	0.563	0.559	0.647	0.647	0.653
TST	0.457	0.448	0.449	0.540	0.539	0.537	0.600	0.603	0.609
MTST	0.450	0.444	0.444	0.535	0.535	0.532	0.595	0.599	0.603
oracle	0.474	0.464	0.467	0.583	0.583	0.583	0.719	0.714	0.717

Table 4.4: Simulation study with independent testing statistics with ”1234” configuration. Estimated values of power for selected values of m and r.

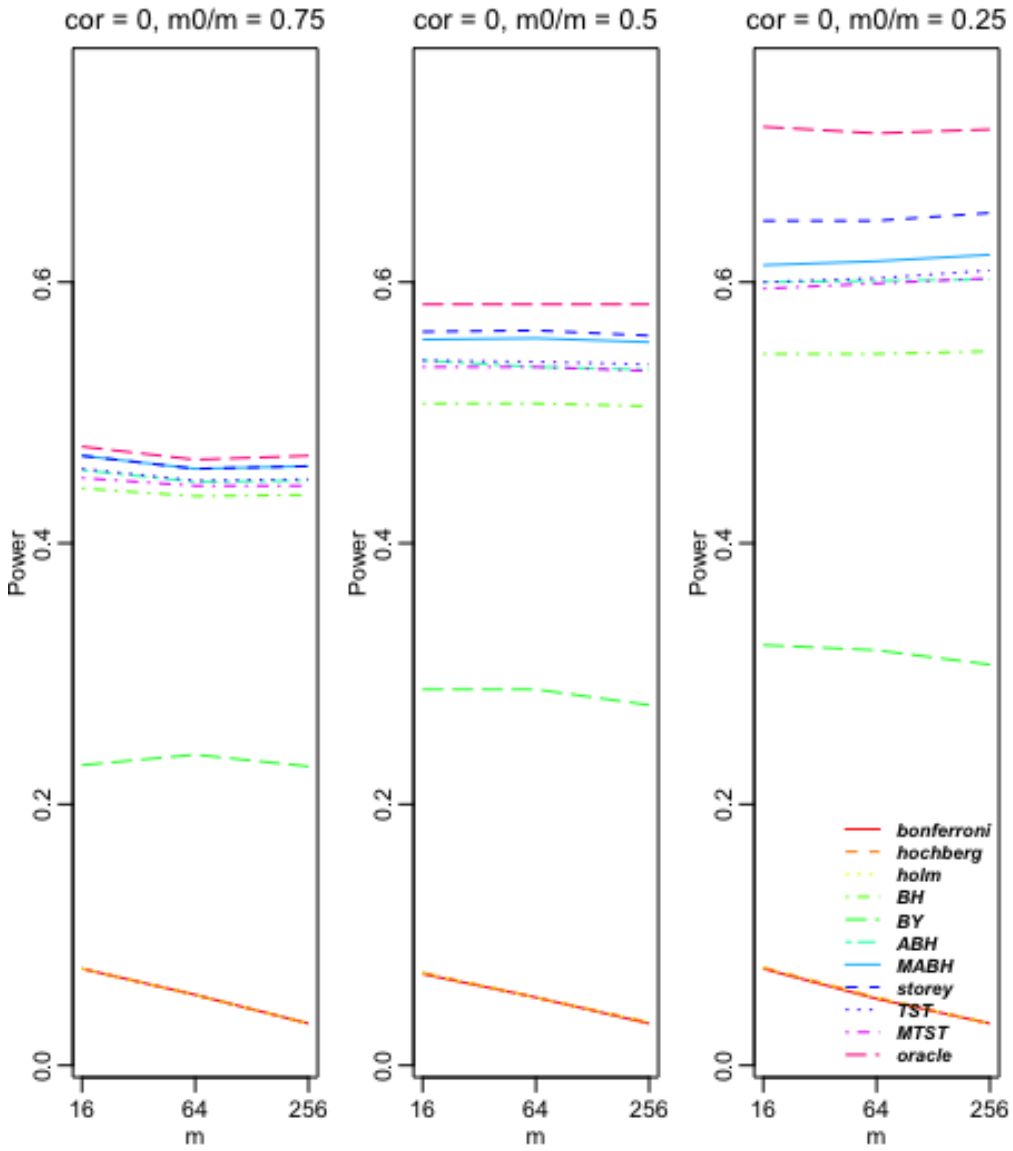

Figure 4.5: Simulation study with independent testing statistics with ”1234” configuration

Naturally, "1234" configuration produces a less drastic difference in p-values between true and false null hypotheses, and thus would lead to lower power compared with the "all-at-5" configuration.

With the decrease in the fraction of true null hypotheses, we observe an increase in power for FDR-controlling methods, but FDR tends to be further below α for all procedures in this case, except for the oracle. With an increasing number of tests, the observation is similar to "all-at-5" configuration.

Under this configuration, we notice a greater variation in power among different procedures. Specifically, when the number of tests is 256 and the fraction of true null hypotheses is 0.25, we could see that the adaptive FDR controlling procedures give power around 20 times higher than FWER controlling methods. The differences between two stage procedures and adaptive procedures are not so obvious.

4.3.2 Positively Dependent Test Statistics

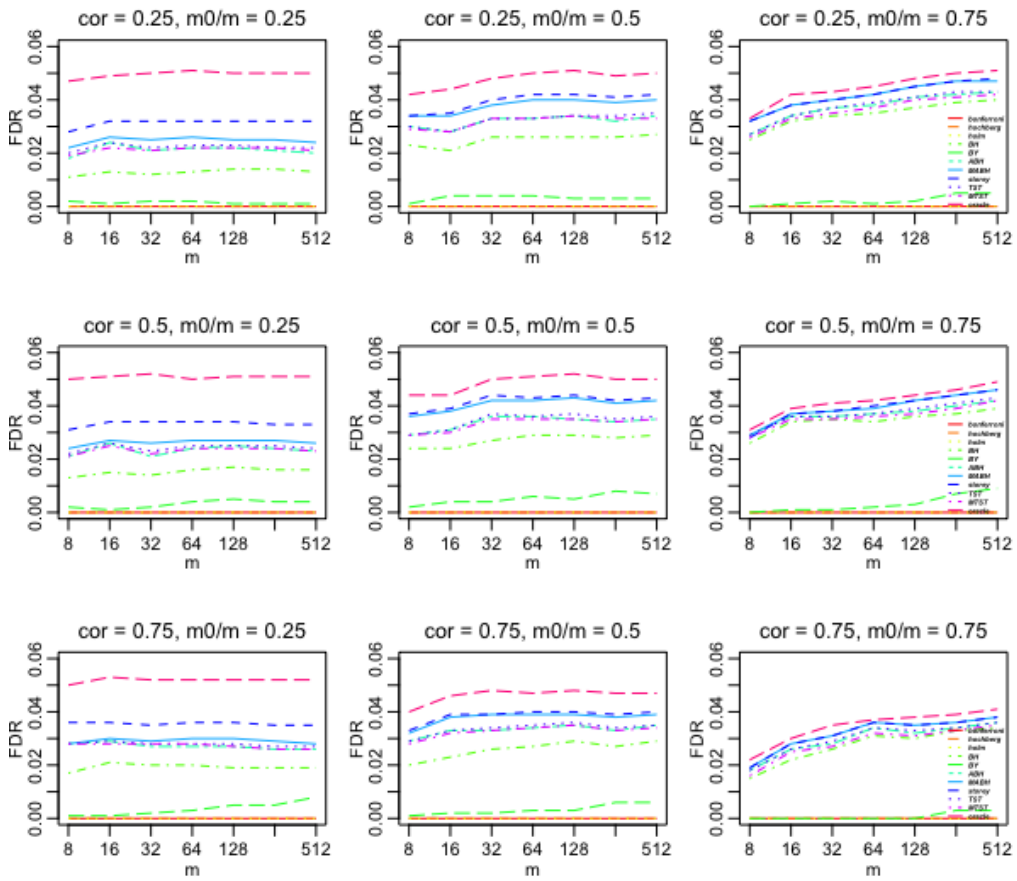

Figure 4.6: Simulation study with independent testing statistics with "1234" configuration

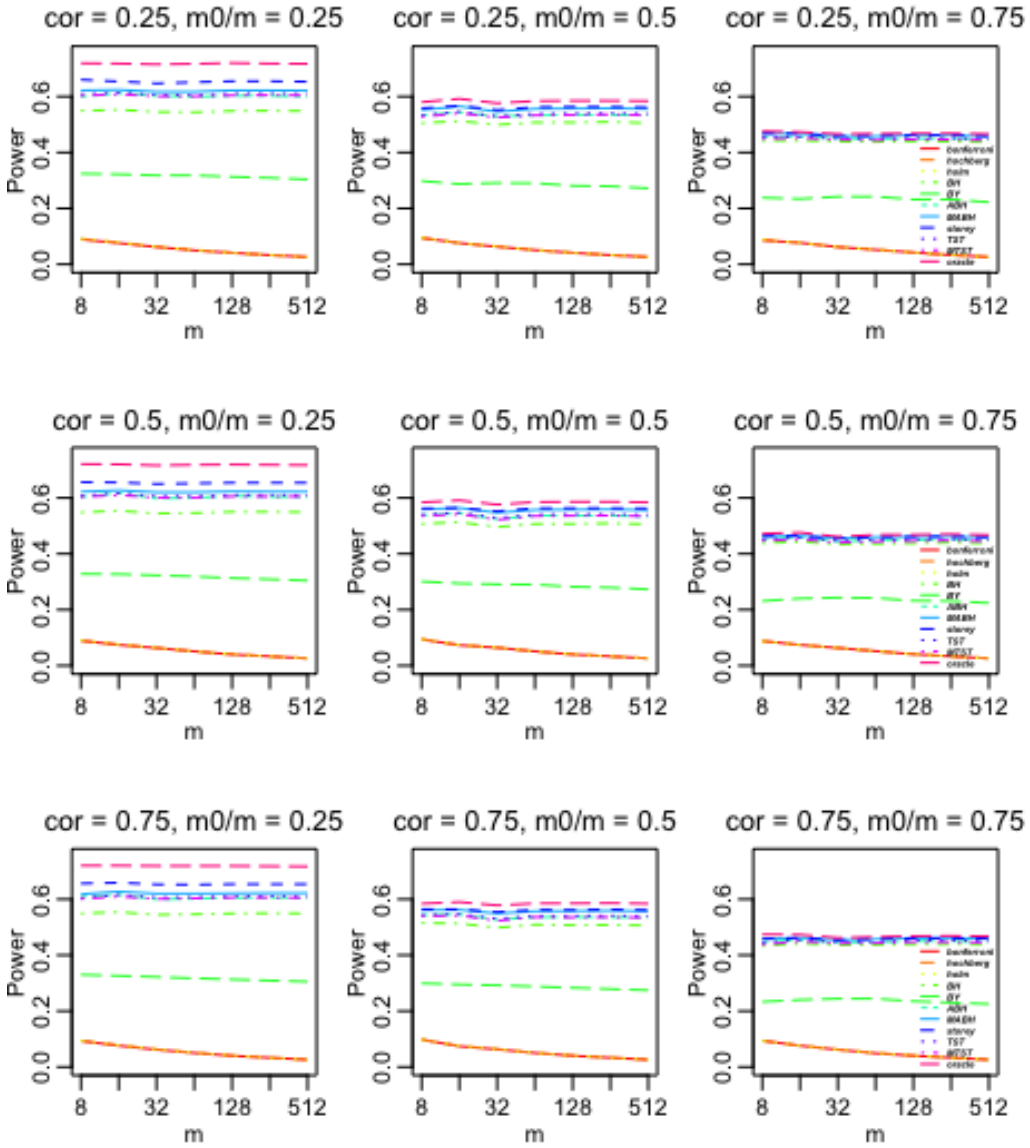

Figure 4.7: Simulation study with independent testing statistics with "1234" configuration

From this result, we could observe that FWER-controlling methods, as well as BH and BY procedures are controlling FDR at a level that is too low. The rest of the methods could control FDR at a level that is close to but below the significance level.

Bibliography

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

Benjamini, Y. and Y. Hochberg (2000). On the adaptive control of the false discovery rate in multiple testing with independent statistics. *Journal of the Educational and Behavioral Statistics 25*, 60–83.

Benjamini, Y., A. Krieger, and D. Yekutieli (2006). Adaptive linear step-up procedures that control the false discovery rate. *Biometrika 93*, 491–507.

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

Efron, N., R. Tibshirani, J. Storey, and V. Tusher (2001). Empirical bayes analysis of a microarray experiment. *Journal of American Statistical Association 96*, 1151–60.

Hochberg, Y. (1995). A sharper bonferroni procedure for multiple tests of significance. *Biometrika 75*, 800–803.

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

J.D. Storey, J.E. Taylor, D. S. (2004). Strong control, conservative point estimation, and simultaneous conservative consistency of false discovery rates: A unified approach. *Journal of the Royal Statistical Society Series B 64*, 187–205.

R. Marcus, P. Eric, K. G. (1976). On closed testing procedures with special reference to ordered analysis of variance. *Biometrika 63*, 655–660.

S.Dudoit and M. V. der Laan (2008). *Multiple testing procedures with applications to genomics*. NY: Springer.

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

Appendix A

R code

```
p.adjust.methods ←
  c("holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr",
    "ABH", "MABH", "storey", "TST", "MTST",  # <<←-- the new methods
    "none")

p.adjust ← function(p, method = p.adjust.methods, n = length(p),
                    alpha = 0.05, lambda = 0.5)
{
  ## Methods 'Hommel', 'BH', 'BY' and speed improvements
  ## contributed by Gordon Smyth
  method ← match.arg(method)
  if(method == "fdr") method ← "BH" # back compatibility
  nm ← names(p)
  p ← as.numeric(p)
  p0 ← setNames(p, nm)
  if(all(nna ← !is.na(p))) nna ← TRUE
  p ← p[nna]
  lp ← length(p)
  stopifnot(n >= lp)
  if (n <= 1) return(p0)
  if (n == 2 && method == "hommel") method ← "hochberg"

  p0[nna] ←
    switch(method,
           bonferroni = pmin(1, n * p),
           holm = {
             i ← seq_len(lp)
             o ← order(p)
             ro ← order(o)
             pmin(1, cummax( (n - i + 1L) * p[o] ))[ro]
           },
           hommel = { ## needs n-1 >= 2 in for() below
             if(n > lp) p ← c(p, rep.int(1, n-lp))
             i ← seq_len(n)
             o ← order(p)
             p ← p[o]
             ro ← order(o)
             q ← pa ← rep.int( min(n*p/i), n)
             for (j in (n-1):2) {
               ij ← seq_len(n-j+1)
               i2 ← (n-j+2):n
               q1 ← min(j*p[i2]/(2:j))
               q[ij] ← pmin(j*p[ij], q1)
               q[i2] ← q[n-j+1]
               pa ← pmax(pa,q)
             }
             pmax(pa,p)[if(lp < n) ro[1:lp] else ro]

           },
```

```
            hochberg = {
              i ← lp:1L
              o ← order(p, decreasing = TRUE)
              ro ← order(o)
              pmin(1, cummin( (n - i + 1L) * p[o] ))[ro]

            },
            BH = {
              i ← lp:1L
              o ← order(p, decreasing = TRUE)
              ro ← order(o)
              pmin(1, cummin( n / i * p[o] ))[ro]
            },

            BY = {
              i ← lp:1L
              o ← order(p, decreasing = TRUE)
              ro ← order(o)
              q ← sum(1L/(1L:n))
              pmin(1, cummin(q * n / i * p[o]))[ro]

            },

            ABH = {
              o ← order(p)
              i ← 1L:lp
              h0.m ← (n+1-i)/(1-p[o])
              idx ← which(diff(h0.m)>0)
              if (length(idx) != 0) {
                m_prev ← h0.m[min(idx) + 1]
                m ← ceiling(min(m_prev, lp))
              }
              else {
                m ← lp
              }

              i ← lp:1L
              o ← order(p, decreasing = TRUE)
              ro ← order(o)
              pmin(1, cummin( m / i * p[o] ))[ro]
            },

            TST = {
              i ← lp:1L
              o ← order(p, decreasing = TRUE)
              ro ← order(o)
              p_adj ← pmin(1, cummin( n / i * p[o] ))[ro]
              m ← sum(p_adj >= alpha/(1+alpha),na.rm=TRUE)
              if (m != 0 & m != n) {
                p_adj ← pmin(1, cummin( m / i * p[o] ))[ro]
              }
              p_adj
            },

            MABH = {
              m ← (n/2)/(1-median(p))
              m ← ceiling(min(m, lp))
              i ← lp:1L
              o ← order(p, decreasing = TRUE)
              ro ← order(o)
              pmin(1, cummin( m / i * p[o] ))[ro]
            },

            storey = {
              r ← sum(p <= lambda, na.rm=TRUE)
              m ← (n - r + 1)/(1 - lambda)
              i ← lp:1L
              o ← order(p, decreasing = TRUE)
              ro ← order(o)
              pmin(1, cummin( m / i * p[o] ))[ro]
```

```
            },

            MTST = {
              i ← lp:1L
              o ← order(p, decreasing = TRUE)
              ro ← order(o)
              p_adj ← pmin(1, cummin( n / i * p[o] ))[ro]
              m ← length(which(p_adj >= alpha))
              if (m != 0 & m != n) {
                p_adj ← pmin(1, cummin( m / i * p[o] * (1+alpha)))[ro]
              }
              p_adj
            },
            none = p)
  p0
}
```

```
library(multtest)
library(sfsmisc)

alpha ← 0.05
n ← 1000

#############
# This function outputs FDR, type 1 error rate, and power,
# given the adjusted_p_value, m, n, and true_frac as input
# It's called inside the sim() function
#############

compute_results← function(adjusted_p,m, n, true_frac, ret){
  adjusted_p ← matrix(adjusted_p, n, m)
  # rejection index
  rejected_idx ← apply(adjusted_p, 1, function(x) which(x < alpha))
  # number of false rejection
  num_false_rejected ← lapply(rejected_idx, function(x) length(which(x <= true_
      frac*m)))
  # number of rejections
  num_rejected ← lapply(rejected_idx, length)

  res ← c()

  # FDR is #false rejection/#rejection
  if ("FDR" %in% ret) {
    num_rejected_adj ← lapply(num_rejected, function(x) max(x,1))
    fdr ← Map('/', num_false_rejected, num_rejected_adj)
    FDR ← round(mean(unlist(fdr)),3)
    res ← c(res, FDR)
  }

  # Type 1 error rate is #false rejection/#true null
  if ("Type_1" %in% ret) {
    type_1 ← lapply(num_false_rejected, function(x) x/(true_frac*m))
    type_1_error_rate ← round(mean(unlist(type_1)), 3)
    res ← c(res, type_1_error_rate)
  }

  # power is #rejected false null/#false null
  if ("Power" %in% ret) {
    num_true_rejected ← Map('-', num_rejected, num_false_rejected)
    pow ← lapply(num_true_rejected, function(x) x/((1-true_frac)*m))
    powers ← round(mean(unlist(pow)),3)
    res ← c(res, powers)
  }
  return(res)
}

##################
# This function takes in the simulation result
# and plots result vs. m
```

```
##################

plot_single ← function(sim_res, m_vals, cor, true_frac, procs = procs, cex=0.4,
    legend=TRUE, ret="fdr", ymin, ymax) {
  colors ← rainbow(dim(sim_res)[1])
  linetype ← c(1:dim(sim_res)[1])

  plot(1,xlim = c(1, dim(sim_res)[2]), ylim = c(ymin, ymax), type='n', xlab = 'm'
      , ylab= ret,xaxt = "n" )

  for (i in 1:dim(sim_res)[1]) {
    lines(c(1:dim(sim_res)[2]), sim_res[i,], col=colors[i], lty=linetype[i])
  }
  axis(1, at=c(1:dim(sim_res)[2]), labels=as.character(m_vals))

  if (legend) {
    legend("bottomright", legend=c(procs,"oracle"),cex=cex, col=colors,lty=
        linetype,text.font=4, bty="n")
    #legend("topleft",legend=c(procs,"oracle"), cex=cex,col=colors, lty=linetype,
         text.font=4, inset=c(1,0), xpd=TRUE, bty="n")
  }
  title(main=sprintf("cor = %s, m0/m = %s", cor, true_frac), font.main=1)
}

##########
# This function outputs the result for selected procedures as a matrix,
# given m, cor, true_frac, and n as input
# We could save the matrix as .csv or .Rdata
##########

sim ← function(cor, true_frac, n, m, procs = c("bonferroni", "holm","hochberg", "
    BH", "BY","ABH", "MABH","storey", "TST", "MTST"), ret = c("FDR", "Type_1", "
    Power"), mu_config = 1, save=FALSE) {
  set.seed(123)
  # generate Z
  Z0 ← matrix(rep(rnorm(n,0,1),m),n,m)
  Z ← matrix(rnorm(n*m, 0, 1), n, m)
  true_len ← as.integer(m*true_frac)

  if (mu_config == 1) {
    # "all at 5" configuration
    mu ← matrix(c(rep(0, true_len*n), rep(5, (m-true_len)*n)), n, m)
  }  else {
    # "1 2 3 4" configuration
    left_mu ← matrix(c(rep(0, true_len*n)), n, true_len)
    right_mu ← t(matrix(rep(c(1,2,3,4), (m-true_len)*n/4), m-true_len,n))
    mu ← cbind(left_mu, right_mu)
  }
  # generate Y
  Y ← sqrt(cor)*Z0 + sqrt(1-cor)*Z + mu
  # generate P
  P ← 2*(1-pnorm(abs(Y)))
  # matrix for storing the result
  mat ← matrix(0,0,length(ret))
  # for each procedure, compute the result
  for(proc in procs) {
    p_vals ← p.adjust(P, method=proc)
    cur_res ← compute_results(p_vals, m, n, true_frac, ret)
    mat ← rbind(mat, cur_res)
  }
  # compute the result for oracle
  p_adj_oracle ← p.adjust(P*true_frac, method="BH")
  oracle_res ← compute_results(p_adj_oracle, m, n, true_frac, ret)
  mat ← rbind(mat, oracle_res)
  rownames(mat) ← c(procs,"oracle")
  colnames(mat) ← ret

  if (save) {
    save_csv = sprintf("mat_m%s_cor%s_frac%s.csv", m, cor, true_frac)
    write.csv(mat, file = save_csv)
```

```
  }
  return(mat)
}


################
# This function takes in a list of cor values, m values, and true fraction values
# the result is stored in a single matrix
# can store the result as .csv or .Rdata
# can plot result vs. m
###############

mult_sim ← function(cor_vals, true_frac_vals, m_vals,ret = c("FDR", "Type_1", "
    Power"), procs = c("bonferroni", "hochberg", "holm", "BH", "BY","ABH", "MABH"
    ,"storey", "TST", "MTST") ,mu_config = 1, save=FALSE, plot=FALSE, ymin=0,
    ymax=0.06, cex=0.4) {

  mat ← matrix(0,length(procs)+1,0)

  for (cor in cor_vals) {

    for (true_frac in true_frac_vals) {

      plot_mat ← matrix(0, length(procs)+1, 0)

      for (m in m_vals) {

        res ← sim(cor,true_frac,n, m, ret =ret, procs = procs, mu_config=mu_
            config)
        plot_mat ← cbind(plot_mat, res)
        mat ← cbind(mat, res)

      }

      if (plot) {
        lg = true_frac == true_frac_vals[length(true_frac_vals)]
        plot_single(plot_mat, m_vals, cor, true_frac, procs, ret=ret, legend=lg,
            ymin=ymin, ymax = ymax, cex=cex)
      }

    }
  }
  if (save) {
    save_csv = sprintf("table_1.csv", cor, true_frac)
    write.csv(mat, file = save_csv)
  }
  return(mat)
}
```

```
---
title: "main"
author: "Tianqi Wang"
date: "12/9/2018"
output: html_document
---
```{r}
# The code below produces p_adj.png
source("p_adjust.R")
p ← 2^seq(-1, -7, by=-0.2)
p.adjust.M ← p.adjust.methods[(p.adjust.methods != "fdr")]
p.adj    ← sapply(p.adjust.M, function(meth) p.adjust(p, meth))

matplot(p, p.adj, ylab="p.adjust", type = "l", asp = 1, lty = 1:12,
        col = 1:12, main = "P-value adjustments", log="xy", xaxt="n", yaxt="n")
legend("bottomright", legend=colnames(p.adj),col=1:12,lty=1:12,cex=0.8, bty="n")
points(p,p, cex=1/4)

sfsmisc::eaxis(1)
sfsmisc::eaxis(2)
```
```

```
21
22 dev.copy(png,'p_adj.png')
23 dev.off()
24 ```
25
26
27 ```{r }
28 # The code below produces f2_indep_fdr.png
29 source("simulation.R")
30
31 cor ← 0
32 true_frac_vals ← c(0.75, 0.5, 0.25)
33 m_vals ← c(16, 64, 256)
34 mult.fig(mfrow=c(1,3), marP = -c(0,1.8,0,0))
35 res_indep_fdr_config_1 ← mult_sim(cor, true_frac_vals, m_vals,ret = "FDR", plot=
       TRUE, cex=0.8, ymax=0.055)
36 res_indep_fdr_config_1
37 ```
38
39
40 ```{r}
41 # The code below produces f3_indep_power.png
42
43 mult.fig(mfrow=c(1,3), marP = -c(0,1.8,0,0))
44 res_indep_power_config_1 ← mult_sim(cor, true_frac_vals, m_vals,ret = "Power",
       plot=TRUE, ymin = 0.4, ymax = 1, cex=0.8)
45 res_indep_power_config_1
46 ```
47
48 ```{r}
49 # The code below produces f4_indep_fdr.png
50 res_indep_fdr_config_2 ← mult_sim(cor, true_frac_vals, m_vals,ret = "FDR", mu_
       config = 2, plot = TRUE, ymax=0.052, cex=0.8)
51 res_indep_fdr_config_2
52 ```
53
54 ```{r}
55 # The code below produces f5_indep_power.png
56 res_indep_power_config_2 ← mult_sim(cor, true_frac_vals, m_vals,ret = "Power", mu
       _config = 2, plot = TRUE, ymin=0.02, ymax=0.75,cex=0.8)
57 res_indep_power_config_2
58 ```
59
60 ```{r}
61 # The code below produces mult_sim_fdr.png
62 mult.fig(mfrow=c(3,3), marP = -c(0,1.8,0,0))
63 m_vals ← c(8,16,32,64,128,256,512)
64 cor_vals ← c(0.25,0.5,0.75)
65 true_frac_vals ← c(0.25,0.5,0.75)
66 res_dep_fdr_config_2 ← mult_sim(cor_vals, true_frac_vals, m_vals,ret = c("FDR"),
       mu_config = 2, plot=TRUE)
67 ```
68
69 ```{r}
70 # The code below produces mult_sim_power.png
71 mult.fig(mfrow=c(3,3), marP = -c(0,1.8,0,0))
72 m_vals ← c(8,16,32,64,128,256,512)
73 cor_vals ← c(0.25,0.5,0.75)
74 true_frac_vals ← c(0.25,0.5,0.75)
75 res_dep_power_config_2 ← mult_sim(cor_vals, true_frac_vals, m_vals,ret = c("Power
       "), mu_config = 2, plot=TRUE, ymin=0., ymax=0.75)
76
77 ```
```

Declaration of Originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor .

Title of work (in block letters):

A COMPARISON ON SEVERAL APPROACHES TO MULTIPLE TESTING PROBLEM

Authored by (in block letters):
For papers written by groups the names of all authors are required.

Name(s):	**First name(s):**

With my signature I confirm that

- I have committed none of the forms of plagiarism described in the Citation etiquette information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work .
- I am aware that the work may be screened electronically for plagiarism.
- I have understood and followed the guidelines in the document *Scientific Works in Mathematics.*

Place, date:

Zürich, 19.12.2018

Signature(s):

Jiangi Wang

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.