DiscreteFDR: An **R** package for controlling the false discovery rate for discrete test statistics

Guillermo Durand
Sorbonne University

Florian Junge
Darmstadt University
of Applied Sciences

Sebastian Döhler
Darmstadt University
of Applied Sciences

Etienne Roquain
Sorbonne University

Abstract

The simultaneous analysis of many statistical tests is ubiquitous in applications. Perhaps the most popular error rate used for avoiding type one error inflation is the false discovery rate (FDR). However, most theoretical and software development for FDR control has focused on the case of continuous test statistics. For discrete data, methods that provide proven FDR control and good performance have been proposed only recently. The R package **DiscreteFDR** (Durand and Junge (2019), version 1.2) provides an implementation of these methods. For particular commonly used discrete tests such as Fisher's exact test, it can be applied as an off-the-shelf tool by taking only the raw data as input. It can also be used for any arbitrary discrete test statistics by using some additional information on the distribution of these statistics. The paper reviews the statistical methods in a non-technical way, provides a detailed description of the implementation in **DiscreteFDR** and presents some sample code and analyses.

Keywords: multiple testing, false discovery rate, package, R, discrete tests, Fisher's exact test.

1. Introduction

Multiple testing procedures are important tools for identifying statistically significant findings in massive and complex data while controlling a specific error rate. An important focus has been given to methods controlling the false discovery rate (FDR), i.e., the expected proportion of falsely rejected hypotheses among all rejected hypotheses, which has become the standard error rate for high dimensional data analysis. Since the original procedure of Benjamini and Hochberg (1995), much effort has been undertaken to design FDR controlling procedures that adapt to various underlying structures of the data, such as the quantity of signal, the signal strength and the dependencies, among others.

The R package **DiscreteFDR**, presented in this paper, deals with adaptation to discrete and non-identically distributed test statics by implementing procedures developed by Döhler, Durand, and Roquain (2018) (in the sequel abbreviated as [DDR]). This type of data arises in many relevant applications, in particular when data represent frequencies or counts. Examples can be found in clinical studies (see e.g., Westfall and Wolfinger (1997)), genome-wide association studies (GWAS) (see e.g., Dickhaus, Straßburger et al. (2012)) and next generation sequencing data (NGS) (see e.g., Chen and Doerge (2015b)). The primary discrete test we have in mind in this paper is Fisher's exact test, see Lehmann and Romano (2006),

but we also sketch an application of **DiscreteFDR** to multiple Poisson tests in the spirit of Jimenez-Otero, de Una-Alvarez, and Pardo-Fernandez (2018).

It is well known (see e.g., Westfall and Wolfinger (1997)) that applying critical values derived for continuous approximations to discrete null distributions can generate a severe power loss, already at the stage of the single tests. A consequence is that using 'blindly' the BH procedure with discrete p values will control the FDR in a too conservative manner. Therefore, more powerful procedures that avoid this conservatism are much sought after in applications, see for instance Karp, Heller et al. (2016), van den Broek, Dijkstra et al. (2015) and Dickhaus, Straßburger et al. (2012).

In the literature, constructing multiple testing procedures that take into account the discreteness of the test statistics has a long history, for more details see [DDR]. The heuristic motivation for the procedures implemented in **DiscreteFDR** is as follows. Let $p_{(1)} \leq \ldots \leq p_{(m)}$ denote the ordered p values and $H_{(1)}, \ldots, H_{(m)}$ the corresponding null hypotheses. The BH procedure [BH] works by rejecting $H_{(1)}, \ldots, H_{(\hat{k})}$, where $\hat{k}$ is the largest integer k such that

$$p_{(k)} \leq \frac{k}{m} \cdot \alpha. \tag{1}$$

Now suppose that the cumulative distribution functions $F_1, \ldots, F_m$ of the p values under the null hypotheses are known and introduce the transformation

$$\xi(t) = \frac{1}{m} \sum_{i=1}^{m} F_i(t), \ \ t \in [0, 1]. \tag{2}$$

For continuous settings we often have $F_i(t) = t$ which implies $\xi(t) = t$ and so we can rephrase (1) as

$$\xi(p_{(k)}) \leq \frac{k}{m} \cdot \alpha. \tag{3}$$

Heyse (2011) proposed to use the transformation ξ in (2), where the F_i need no longer be uniform and identical. The benefit of this approach is that - depending on the discreteness and heterogeneity of the involved p value distributions - $\xi(t)$ may be much smaller than t. Clearly, the smaller the ξ-values, the more hypotheses can be rejected. Figure 1 displays such a function where the functions $F_1, \ldots, F_{2446}$ are derived from $m = 2446$ independent Fisher's exact test statistics based on the pharmacovigilance data from Heller and Gur (2011) (see Section 5 for more details). In this example we have $\xi(t) \approx t/3$, thereby yielding a potentially strong rejection enhancement.

Unfortunately, the Heyse procedure does not rigorously control the FDR in general; counter-examples are provided in Heller and Gur (2011) and [DDR]. To correct this, [DDR] introduce new procedures relying on the following modifications of the ξ function (more details are presented in Section 3):

$$\xi_{\mathrm{SU}}(t) = \frac{1}{m} \sum_{i=1}^{m} \frac{F_i(t)}{1 - F_i(\tau_m)}; \ \ \xi_{\mathrm{SD}}(t) = \frac{1}{m} \sum_{i=1}^{m} \frac{F_i(t)}{1 - F_i(t)}, \ \ t \in [0, 1],$$

where τ_m is the generalized inverse of ξ_{SD} taken at point α. Figure 1 demonstrates that the difference between these modifcations and the original ξ can be very small, in particular

for small values of t. In addition, [DDR] also introduce more powerful 'adaptive' versions, meaning that the derived critical values are designed in a way that 'implicitly estimates' the overall proportion of true null hypotheses. All these procedures provide rigorous FDR control under independence of the p values and are implemented in the R package **DiscreteFDR**.
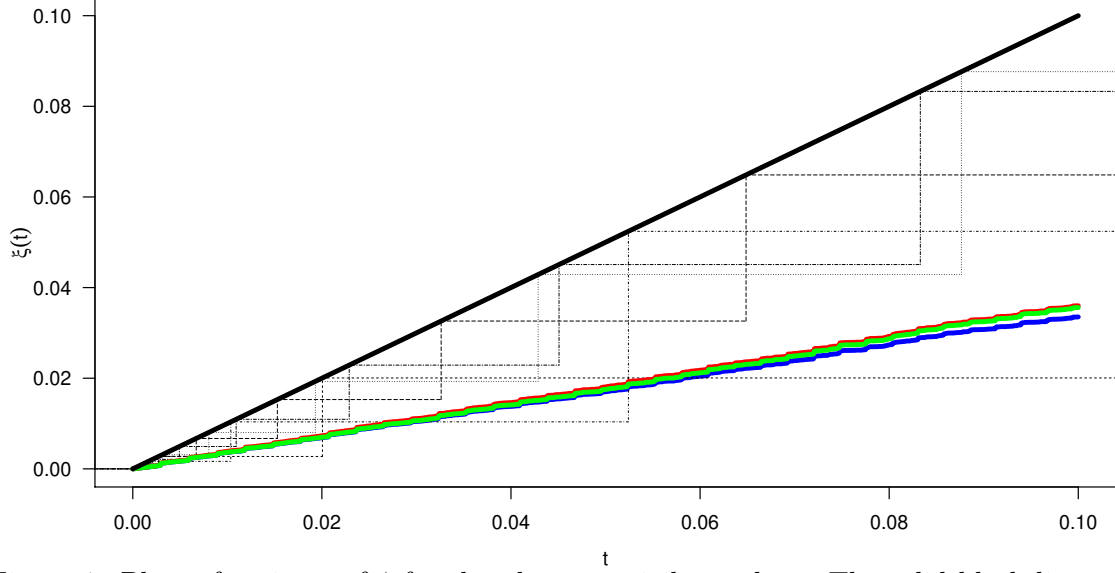

Figure 1: Plots of variants of ξ for the pharmacovigilance data. The solid black line corresponds to the uniform case, the discrete variants are represented by blue (for ξ), green (for ξ_{SD}) and red (for ξ_{SU}) solid lines. Additionally, five arbitrarily selected F_i's are displayed by using different line types.

While there exist numerous R functions and packages that implement multiple testing procedures in the continuous setting (see e.g., Hothorn, Bretz, and Westfall (2008) and Blanchard, Dickhaus et al. (2017)), there are only relatively few tools available that deal specifically with discrete test statistics. The package **MHTdiscrete** (see Zhu and Guo (2017)) is described by its authors as a 'comprehensive tool for almost all existing multiple testing methods for discrete data'. It implements several FWER and FDR procedures designed for discrete data. While the procedures for FWER control are extensively described in an accompanying preprint (see Zhu and Guo (2017)), there seems to be no detailed mathematical description of the implemented FDR procedures. The package **discreteMTP** (see Heller, Gur, and Yaacoby (2012)) also implements several methods aiming at FDR control (including the Heyse procedure) described in more detail in Heller and Gur (2011). The main contribution of the package **DiscreteFDR** is to provide practitioners with a simple to use set of tools (both adaptive and non-adaptive) for analysing discrete data with both proven FDR control and good performance.

In this paper, our primary aim is to introduce **DiscreteFDR**. As an 'appetizer', we start by illustrating the main ideas through analysis of a toy data set. We hope to convince readers that it is worthwile to use discrete FDR methods. We then review the mathematical methods and results from [DDR], followed by some more technical details of the implementation in Section 4 . Section 5 contains an analysis of some real data and includes an example that illustrates how **DiscreteFDR** can be used for arbitrary discrete tests. The paper concludes with a summary and discussion.

We realize - and indeed hope - that the audience of this paper may be quite heterogeneous,

which is why we would like to suggest some guidance for possible ways of reading it. For subject matter scientists and practitioners who may not be interested in the mathematical or software details, we especially recommend to study Sections 2 and 5. For readers who additionally want to understand more of the mathematical background we recommend Section 3, for readers interested in the implementation details of the R-package we recommend Section 4.

2. A toy example

To give a first impression of how **DiscreteFDR** works, we consider an artifical toy example. A more realistic example involving pharmacovigilance data is given in Section 5.

Suppose we would like to compare two treatments in nine different populations. For each population we do this by evaluating the responders and non-responders for each treatment. This leads to categorical data which can be represented, for each population $i = 1, \ldots, 9$ in the following 2×2 table:

	responders	non responders	total
treatment 1	x_{1i}	y_{1i}	n_{1i}
treatment 2	x_{2i}	y_{2i}	n_{2i}
total	$x_{1i} + x_{2i}$	$y_{1i} + y_{2i}$	$n_i = n_{1i} + n_{2i}$

Table 1: 2×2 table for population i.

Denoting the responder probabilities for population i by π_{1i} and π_{2i} we can test e.g.

$$H_{0i} : \pi_{1i} = \pi_{2i} \qquad \text{vs.} \qquad H_{1i} : \pi_{1i} \neq \pi_{2i}$$

by using Fisher's (two-sided) exact test (see Lehmann and Romano (2006), which is implemented in the R function `fisher.test`). Suppose the data in the nine populations are independent and we observe the following data frame `df`

```
   X1  Y1 X2  Y2
1   4 144  0 132
2   2 146  0 132
3   2 146  1 131
4  14 134  3 129
5   6 142  2 130
6   9 139  1 131
7   4 144  2 130
8   0 148  2 130
9   1 147  2 130
```

In this data frame each of the 9 rows represents the data of an observed 2×2 table: e.g., the third row of the data corresponds to $x_{13} = 2, y_{13} = 146, x_{23} = 1, y_{23} = 131$. Even though in this example, the total number of tested hypotheses $m = 9$ is very small, for illustrative purposes we deal with the multiplicity problem here by controlling FDR at level $\alpha = 5\%$.

The DBH step-down procedure (to be explained in more detail in Section 3) can be applied directly to the data frame object `df` and yields the following adjusted p values:

```
R> library("DiscreteFDR")
R> DBH.sd.fast <- fast.Discrete(df, alternative = "two.sided",
+                                  direction = "sd")
R> DBH.sd.fast$Adjusted

[1] 0.25630985 1.00000000 1.00000000 0.03819796 0.51482782 0.03819796
[7] 1.00000000 0.47895996 1.00000000
```

Thus we can reject two hypotheses at FDR-level $\alpha = 5\%$. In order to compare this with the usual (continuous) BH procedure we have to determine the raw p values first. This would be possible by applying the `fisher.test` function to all nine 2×2 tables. Alternatively, we may use the more convenient function `fisher.pvalues.support` included in our package for accessing the raw p values:

```
R> p <- fisher.pvalues.support(df, alternative = "two.sided")
R> raw.pvalues <- p$raw
R> p.adjust(raw.pvalues, method = "BH")

[1] 0.37430072 0.74976959 1.00000000 0.09570921 0.51928737 0.09570921
[7] 0.77313633 0.49804147 0.77313633
```

Applying the continuous BH procedure from the stats package in the last line of code, we find that we can not reject any hypotheses at FDR-level $\alpha = 5\%$. As this example illustrates, the discrete approach can potentially yield a large increase in power. The gain depends on the involved distribution functions and the raw p values. To appreciate where this comes from, it is instructive to consider the distribution functions $F_1, \ldots, F_9$ of the p values under the null in more detail. Take for instance the first 2×2 table:

	responders	non responders	total
treatment 1	4	144	148
treatment 2	0	132	132
total	4	276	280

Table 2: 2×2 table for population 1.

Fisher's exact test works by determining the probability of observing this (or a more 'extreme') table, given that the margins are fixed. So each F_i is determined by the margins of table i. Since $x_{11} + x_{21} = 4$, the only potentially observable tables are given by $x_{11} = 0, \ldots, 4$. For each one of these 5 values a p value can be determined using the hypergeometric distribution. Therefore, the p value of any 2×2 table with margins given by the above table can take (at most) 5 distinct values, say $x_1, \ldots, x_5$. Combining these 5 values into a set, we obtain the *support* $\mathcal{A}_1 = \{x_1, \ldots, x_5\}$ of distribution F_1. Now we can continue in this vein for the remaining 2×2 tables to obtain the supports $\mathcal{A}_1, \ldots, \mathcal{A}_9$ for the distributions functions $F_1, \ldots, F_9$. The supports can be accessed via the `$support` command, e.g.

```
R> p$support[c(1,5)]
```

```
[[1]]
[1] 0.04820493 0.12476691 0.34598645 0.62477763 1.00000000

[[2]]
[1] 0.002173856 0.007733719 0.028324482 0.069964309 0.154043258
[6] 0.288492981 0.481808361 0.726262402 1.000000000
```

returns $\mathcal{A}_1$ and $\mathcal{A}_5$. Panel (a) in Figure 2 shows a graph of the distribution functions $F_1, \ldots, F_9$. Each F_i is a step-function with $F_i(0) = 0$, the jumps occuring only on the support $\mathcal{A}_i$ and $F_i(x) = x$ only for $x \in \mathcal{A}_i$. In particular, all distributions are stochastically larger than the uniform distribution (i.e., $F_i(x) \leq x$), but in a heterogeneous manner. This heterogeneity can be exploited e.g., by transforming the raw p values from the exact Fisher's test using the function $\xi_{\text{SD}}(x) = \sum_{i=1}^{9} \dfrac{F_i(x)}{1 - F_i(x)}$ presented in the Introduction. Panel (b) shows that ξ_{SD} is a step function. Its jumps occur on the joint support $\mathcal{A} = \mathcal{A}_1 \cup \ldots \cup \mathcal{A}_9$. Panel (b) also shows that $\xi_{\text{SD}}(x) \ll x$, at least for small values of x. It turns out that the critical values of our new DBH step-down procedure are essentially given by inverting ξ_{SD} at the critical values of the [BH] procedure $1 \cdot \alpha/9, 2 \cdot \alpha/9, \ldots, \alpha$, so that these values are considerably larger than the [BH] critical values (for more details see Section 3). This is illustrated in panel (c) along with the ordered p values. In particular, all asterisks are located above the green [BH] dots, therefore this procedure can not reject any hypothesis. In contrast, the two smallest p values are located below red DBH step-down dots, so that this procedure rejects two hypotheses as we had already seen earlier.
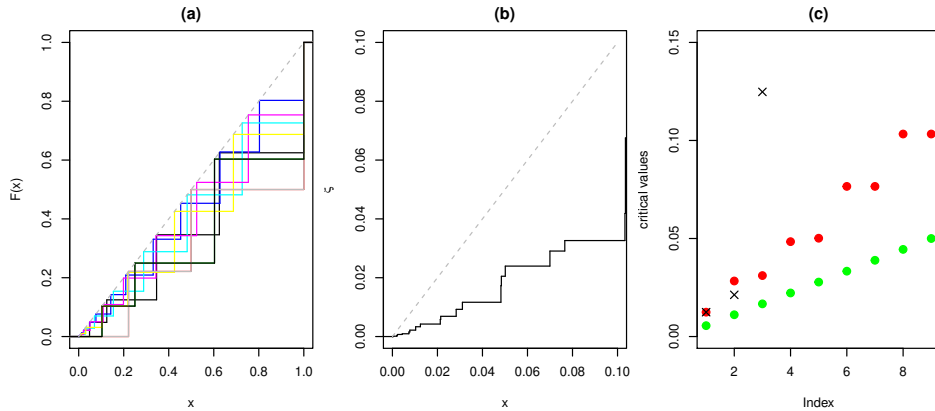

Figure 2: Panel (a) depicts the distribution functions $F_1, \ldots, F_9$ in various colours, (b) is a graph of the transformation ξ_{SD}. The uniform distribution function is shown in light grey in (a) and (b). Panel (c) shows the [BH] critical values (green dots), the DBH step-down critical values (red dots) and the sorted raw p values (asterisks).

3. Implemented FDR-controlling procedures

The procedures used in the package are all based upon a comparison between the ordered p values $p_{(k)}$, $1 \leq k \leq m$, and a sequence of nondecreasing *critical values* τ_k, $1 \leq k \leq m$. Depending on how these two sequences intercept allow to define a rejection number $\hat{k}$ and thus a rejection set of null hypotheses $H_{(1)}, \ldots, H_{(\hat{k})}$. Classically, the *step-up procedure* with critical values τ_k, $1 \leq k \leq m$, corresponds to choose the last right crossing point

$$\hat{k}_{SU} = \max\{k \, : \, p_{(k)} \leq \tau_k\}.$$

Hence, it goes backwards, starting from the largest p value $p_{(m)}$, stopping the first time it finds k_0 such that $p_{(k_0)} \leq \tau_{k_0}$ and returning $\hat{k}_{SU} = k_0$. By contrast, the *step-down procedure* with critical values τ_k, $1 \leq k \leq m$ uses the first left crossing point

$$\hat{k}_{SD} = \max\{k \, : \; \text{for all } k' \leq k, \, p_{(k')} \leq \tau_{k'}\}.$$

Hence, it goes forward, starting from the smallest p value $p_{(1)}$, stopping the first time it finds k_0 such that $p_{(k_0)} > \tau_{k_0}$ and returning $\hat{k}_{SD} = k_0 - 1$.

Such multiple testing procedures are thus driven by a sequence of critical values and by a choice between the step-up or step-down version. In our package, the 5 different possible choices are listed in Table 3, with 3 step-up procedures [DBH-SU], [A-DBH-SU], [DBR-λ] and 2 step-down procedures [DBH-SD], [A-DBH-SD]. We easily check that [A-DBH-SU] (resp. [A-DBH-SD]) rejects always more null hypotheses than [DBH-SU] (resp. [DBH-SD]). Note that the names of the procedures are slightly different in the original paper [DDR]. This is done to emphasize that our package is primarily devoted to the discrete case.

3.1. Critical values

The specific shape of the critical values comes from the FDR upper-bounds derived in [DDR], which ensures that these procedures control the FDR at the nominal level α under independence of the p values, see Theorem 1 and Corollary 1 in [DDR]. In Table 3, each F_i is defined as the (least favorable) cumulative distribution function of the p value p_i under the null hypothesis. As in the example from Section 2, $\mathcal{A} = \mathcal{A}_1 \cup \ldots \cup \mathcal{A}_m \subset [0, 1]$ stands for the union of the supports of the marginal distributions of the p values, p_i, $1 \leq i \leq m$ which can be determined under the full hull hypothesis, i.e., when all null hypotheses are assumed to be true. While $\mathcal{A} = [0, 1]$ in the case where the F_i's are continuous functions, the primary setting we have in mind is a large number of simultaneous Fisher exact tests, so that $\mathcal{A} = \mathcal{A}_1 \cup \ldots \cup \mathcal{A}_m$ is finite but very large. See also Section 2 for some concrete examples.

Let us underline that obtaining such τ_k numerically might be time consuming because the overall support $\mathcal{A}$ can be large while testing whether each $t \in \mathcal{A}$ satisfies the required condition given by the second column in Table 3 involves a complex combination of the F_i, $1 \leq i \leq m$. In the package, we have implemented a shortcut that reduces the range of $t \in \mathcal{A}$ that has to be explored: it is based on the fact that if $F_i(t) \leq t$ for all t and i (super-uniformity), we have the following lower bounds $\tau_k^{\min}$'s on the critical values τ_k's, see Lemmas 2, 3 and 4 in [DDR]:

[DBH-SU]	$\tau_k^{\min} = \max\{t \in \mathcal{A} \ : \ t \leq \alpha k/m(1+\alpha)^{-1}\}$
[DBH-SD]	$\tau_k^{\min} = \max\{t \in \mathcal{A} \ : \ t \leq \alpha k/m(1+\alpha k/m)^{-1}\}$
[A-DBH-SU]	$\tau_m^{\min} = \max\{t \in \mathcal{A} \ : \ t \leq \alpha(1+\alpha)^{-1}\}$
	$\tau_k^{\min} = \max\left\{t \in \mathcal{A} \ : \ t \leq \tau_m \wedge \left((1-\tau_m)\dfrac{\alpha k}{m-k+1}\right)\right\}, \ k < m;$
[A-DBH-SD]	$\tau_k^{\min} = \max\{t \in \mathcal{A} \ : \ t \leq \alpha k/(m-(1-\alpha)k+1)\}$
[DBR-λ]	$\tau_k^{\min} = \max\left\{t \in \mathcal{A} \ : \ t \leq \lambda \wedge \left((1-\lambda)\dfrac{\alpha k}{m-k+1}\right)\right\}$

Our current implementations of [DBH-SU] and [A-DBH-SU] first determine τ_m by searching for τ_m in $\mathcal{A} \cap [\tau_m^{\min}, 1]$ and then determine all other τ_k simultaneously using $\tau_k \geq \tau_1^{\min}$ instead of $\tau_k \geq \tau_k^{\min}$. We take this approach for simplicity and performance reasons. The stepdown procedures only use the latter constraint. These lower bounds help to reduce the computational burden considerably.

3.2. Transformed p values

If we are only interested in the set of rejected null hypotheses and not in the critical values, we can significantly speed-up the program by skipping the explicit computation of the critical values and by directly considering the transformed p values:

$$p'_k = \xi_k(p_{(k)}), \ 1 \leq k \leq m, \tag{4}$$

where the functions $\xi_k(\cdot)$, defined in Table 3, are such that τ_k is the inverse of ξ_k at point $\alpha k/m$. Note that while the elements of $\{p_{(k)}, 1 \leq k \leq m\}$ are ordered, this is not necessarily the case for the elements of $\{p'_k, 1 \leq k \leq m\}$. The following proposition is obvious.

Proposition 1 *For each of the critical values listed in Table 3 we have for all $1 \leq k \leq m$,*

$$p_{(k)} \leq \tau_k \iff p'_k \leq \alpha k/m, \tag{5}$$

where p'_k are the transformed p values defined by (4).

A consequence of (5) is that the step-up and step-down cutoffs can be computed by using only the transformed p values and the BH critical values as follows:

$$\hat{k}_{SU} = \max\{k \ : \ p'_k \leq \alpha k/m\}$$
$$\hat{k}_{SD} = \max\{k \ : \ \text{for all } k' \leq k, \ p'_{k'} \leq \alpha k'/m\}.$$

Thus, all of the above methods can be interpreted as variant of the classical (SU or SD) BH procedure, for which each p value has been suitably transformed to account for discreteness.

3.3. Adjusted p values

In applications, it is often convenient for the analyst to use *adjusted p values* $\widetilde{p}_k$ instead of the raw p values and rejecting those hypotheses for which $\widetilde{p}_k \leq \alpha$. The advantage of this approach is that it is more convenient to apply and easier to communicate. Furthermore, it avoids to explicitly rely on the, often somewhat arbitrary, choice of α. With the transformations

Procedure	Critical values	Transformation
[DBH-SU]	$\tau_m = \max\left\{ t \in \mathcal{A} : \dfrac{1}{m}\sum_{i=1}^{m} \dfrac{F_i(t)}{1-F_i(t)} \leq \alpha \right\}$	$\xi_m(t) = \dfrac{1}{m}\sum_{i=1}^{m} \dfrac{F_i(t)}{1-F_i(t)}$
$k < m$	$\tau_k = \max\left\{ t \in \mathcal{A} : t \leq \tau_m,\ \dfrac{1}{m}\sum_{i=1}^{m} \dfrac{F_i(t)}{1-F_i(\tau_m)} \leq \alpha k/m \right\}$	$\xi_k(t) = \begin{cases} \dfrac{1}{m}\sum_{i=1}^{m} \dfrac{F_i(t)}{1-F_i(\tau_m)} & ,\, t \leq \tau_m \\[2mm] 1 & ,\, \text{else} \end{cases}$
[DBH-SD]	$\tau_k = \max\left\{ t \in \mathcal{A} : \dfrac{1}{m}\sum_{i=1}^{m} \dfrac{F_i(t)}{1-F_i(t)} \leq \alpha k/m \right\}$	$\xi_k(t) = \dfrac{1}{m}\sum_{i=1}^{m} \dfrac{F_i(t)}{1-F_i(t)}$
[A-DBH-SU]	$\tau_m = \max\left\{ t \in \mathcal{A} : \dfrac{1}{m}\sum_{i=1}^{m} \dfrac{F_i(t)}{1-F_i(t)} \leq \alpha \right\}$	$\xi_m(t) = \dfrac{1}{m}\sum_{i=1}^{m} \dfrac{F_i(t)}{1-F_i(t)}$
$k < m$	$\tau_k = \max\left\{ t \in \mathcal{A} : t \leq \tau_m,\ \sum_{\ell=1}^{m-k+1}\left(\dfrac{F(t)}{1-F(\tau_m)} \right)_{(\ell)} \leq \alpha k \right\}$	$\xi_k(t) = \begin{cases} \dfrac{1}{m}\sum_{\ell=1}^{m-k+1}\left(\dfrac{F(t)}{1-F(\tau_m)} \right)_{(\ell)} & ,\, t \leq \tau_m \\[2mm] 1 & ,\, \text{else} \end{cases}$
[A-DBH-SD]	$\tau_k = \max\left\{ t \in \mathcal{A} : \sum_{\ell=1}^{m-k+1}\left(\dfrac{F(t)}{1-F(t)} \right)_{(\ell)} \leq \alpha k \right\}$	$\xi_k(t) = \dfrac{1}{m}\sum_{\ell=1}^{m-k+1}\left(\dfrac{F(t)}{1-F(t)} \right)_{(\ell)}$
[DBR-λ]	$\tau_m = \max\left\{ t \in \mathcal{A} : (F(t))_{(1)} \leq ((1-\lambda)m\alpha) \wedge \lambda \right\}$	$\xi_m(t) = \begin{cases} \dfrac{(F(t))_{(1)}}{m(1-\lambda)} & ,\, (F(t))_{(1)} \leq \lambda \\[2mm] 1 & ,\, \text{else} \end{cases}$
$k < m$	$\tau_k = \max\left\{ t \in \mathcal{A} : (F(t))_{(1)} \leq \lambda,\ \sum_{\ell=1}^{m-k+1}(F(t))_{(\ell)} \leq \alpha k(1-\lambda) \right\}$	$\xi_k(t) = \begin{cases} \dfrac{\sum_{\ell=1}^{m-k+1}(F(t))_{(\ell)}}{m(1-\lambda)} & ,\, (F(t))_{(1)} \leq \lambda \\[2mm] 1 & ,\, \text{else} \end{cases}$

Table 3: Implemented procedures (left column), critical values (center) and associated transformation functions (right column). The suffix 'SU' stands for step-up, the suffix 'SD' for step-down procedures.

introduced above, it is straightforward to define (variants of) discrete FDR-adjusted p values. The generic definition given in Dudoit and van der Laan (2007) then yields

$$\widetilde{p}_k = \min_{\ell=k,\ldots,m} \left(\frac{m}{\ell} \cdot p'_\ell \right) \wedge 1, \quad 1 \le k \le m \tag{6}$$

for step-up procedures and

$$\widetilde{p}_k = \max_{\ell=1,\ldots,k} \left(\frac{m}{\ell} \cdot p'_\ell \right) \wedge 1, \quad 1 \le k \le m, \tag{7}$$

for step-down procedures. For our step-down procedures, the usual result holds true.

Proposition 2 *For the step-down procedures [DBH-SD], [A-DBH-SD] and the step-up procedure [DBR-λ] listed in Table 3 we have for all $\alpha \in (0,1)$, for all $1 \le k \le m$,*

$$H_{(k)} \text{ is rejected by the procedure taken at level } \alpha \iff \widetilde{p}_k \le \alpha,$$

where $\widetilde{p}_k$ are the adjusted p values defined by (7).

In the above proposition, note that $H_{(k)}$ is given by the original ordering of the p values $\{p_i, 1 \le i \le m\}$.

For the procedures [DBH-SU], [A-DBH-SU], the situation is more complicated since the adjusted p value $\widetilde{p}_k$ depends on α (through τ_m). The statement in Proposition 2 actually still holds in that situation but not the usual interpretation that the adjusted p value $\widetilde{p}_k$ is the smallest level α at which the procedure rejects $H_{(k)}$. Hence, the analyst would need to exercise care in interpreting them. To avoid any confusion, the package does not report adjusted p values for [DBH-SU] and [A-DBH-SU].

4. Implementation in the package DiscreteFDR

4.1. General structure

The package consists of four groups of functions:

Main functions	`discrete.BH`
	`DBR`
Kernel functions	`kernel.DBH.crit`
	`kernel.DBH.fast`
	`kernel.ADBH.crit`
	`kernel.ADBH.fast`
	`kernel.DBR.crit`
	`kernel.DBR.fast`
Helper functions	`match.pvals`
	`build.stepfuns`
	`short.eff`
	`fisher.pvalues.support`
Wrapper functions	`DBH`
	`ADBH`
	`fast.discrete`

The `discrete.BH` function implements [DBH-SU], [DBH-SU], [A-DBH-SU] and [A-DBH-SD]. Similarly, `DBR` implements [DBR-λ]. They use the first three of the helper functions for common operations (see details in 4.2) and the kernels for the actual computation. [DBH-SU], [DBH-SD], [A-DBH-SU] and [A-DBH-SD] can be accessed directly through the wrapper functions `DBH` and `ADBH`, respectively.

The wrapper function `fast.discrete` applies the Discrete FDR-controlling procedures, which are implemented in `discrete.BH`, to a set of 2×2 contingency tables, given by a matrix or data frame. It uses the `fisher.pvalues.support` helper function to compute p value c.d.f.s and raw p values from these tables in the framework of Fisher's exact test.

We also provide the `amnesia` data set, used in our examples in Section 5 and in our paper [DDR]. It is basically the amnesia data set of package **discreteMTP**, but slightly reformatted (the difference lies in the third column).

The end user should only use the main functions `DBR` and `discrete.BH`, and the wrapper functions `fast.discrete`, `DBH` and `ADBH`. The other functions are only internal functions called by the main ones. We intentionally did not hide them, so that interested users would be able to understand how the main procedures work.

The functions `discrete.BH`, `DBH`, `ADBH` and `DBR` take the following input values:

`raw.pvalues`	The vector (of the same length as `pCDFlist`) of raw observed p values which is calculated from the data.
`pCDFlist`	A list of vectors that represent the supports $\mathcal{A}_1, \ldots, \mathcal{A}_m$ of the discrete distribution functions $F_1, \ldots, F_m$ under the respective null hypotheses, as described in Section 3.
`alpha`	The global significance level $\alpha \in (0, 1)$ at which the procedure provides FDR control; the default is 0.05.
`direction`	(`DBH` and `ADBH` only) A string, either `"su"` or `"sd"`, specifying whether the step-up variant (the default, `"su"`) or the step-down variant (`"sd"`) should be used.
`adaptive`	Specifying whether the adaptive version is to be used (`TRUE`) or not (`FALSE`).
`lambda`	(`DBR` only) The λ parameter of the [DBR-λ] procedure as in Table 3; the default is 0.05.
`ret.crit.consts`	Specifying whether the critical values τ_k are to be computed and included in the output list, at the expense of computational speed; the default is `FALSE`.

They provide the following outputs:

`Rejected`	A vector containing the rejected raw p values
`Indices`	A vector containing the indices of rejected hypotheses
`k.hat`	Number of rejected hypotheses. This corresponds to $\hat{k}_{SU}$ and $\hat{k}_{SD}$, as described in Section 3.2.
`Alpha`	Maximum significance level for the transformed p values for which a rejection occurred, that is `Alpha` $= \alpha \cdot$ `k.hat`$/m$. This corresponds to τ_k $(k = 1, \ldots, m)$ as in Section 3.
`Critical.constants`	A vector containing the critical values (if requested)

`Adjusted`	A vector containing adjusted p values (if available)
`Lambda`	(`DBR` only) The parameter `lambda` that was used when calling `DBR`

More details as to the implementation are provided in the following part.

4.2. Details for some specific functions

Helper functions

The `match.pvals` function performs nearest-neighbor matching for all elements of raw.pvalues, i.e., it checks for each value whether it occurs in its respective p value c.d.f. If this is not the case, it is replaced by the value that is closest to it, that is, its nearest neighbor in its c.d.f. This is to ensure that all values of raw.pvalues actually originate from their respective c.d.f.s, e.g., to correct rounding errors. It has been inspired by a help page of the package **discreteMTP**.

`build.stepfuns` converts the vectors in `pCDFlist` to step function objects. This makes them easier to evaluate in the kernel functions. It is assumed and required that $F_i(t) \leq t$ applies to all c.d.f.s. Compliance with this premise cannot be not checked, so the user is responsible for providing correct vectors. If this condition is not met, the results may be incorrect.

The `short.eff` function extracts all values from a sorted vector that are greater than or equal to the effective critical value associated to a threshold. It simply replaces multiple recurring lines of code with one single function call.

`fisher.pvalues.support` computes discrete raw p values and their support for the test of no association between two categorical variables in 2 x 2 contingency tables using Fisher's exact tests. The p values are computed directly by `phyper`, instead of `fisher.test`, because the latter is much slower. The function is used by the `fast.discrete` function to apply such contingency tables directly to the `discrete.BH` function.

Main Functions

Basically, both main functions have the same workflow:

1. Use `match.pvals` for matching of raw p values with the c.d.f.s and sort the results in ascending order.

2. Convert the c.d.f. vectors in `pCDFlist` to step functions by means of `build.stepfuns`.

3. Determine the overall support $\mathcal{A} = \bigcup \mathcal{A}_i$ from the individual c.d.f.s, remove double values and sort them in ascending order.

4. Use the knowledge of lower bounds, as presented in Section 3.1, to remove unnecessary elements from the support.

5. Compute transformed p values and/or critical values (if requested) with kernel functions (see 4.2.3).

6. Create output list with elements as described in Section 4.1.

Kernel functions

As stated in Section 3, there are two ways to determine which hypotheses corresponding to the elements of a raw p value vector can be rejected.

1. With critical values (see 3.1): this approach works by first determining the critical values. Especially when the size of the support and the number of hypotheses are large, it is computationally intensive, because all elements of the support have to be evaluated by every single c.d.f.

2. With transformed p values (see Section 3.2): here, only the raw p values are evaluated. Thus, it is much more efficient.

As a result, all three implemented procedures have two kernels, that is, a fast one for simplified computation and a slower implementation that calculates critical values. These values are then used to determine which hypotheses are to be rejected and which are not.

The kernel functions need the following parameters:

`stepf`	A list of step function objects.
`pv.numer`	A vector of values from the support for the argument of the F_i in the numerators of the fractions in the formulas presented in Table 3.
`pv.denom`	A vector of p values or a single one for the denominators as in Table 3.

The critical values kernels additionally need:

`alpha`	A numeric value specifying the global significance level.
`sorted.pv`	A numeric vector of observed p values in ascending order.

For the [DBH] and [ADBH] procedures, a direction (`"su"` for step-up or `"sd"` for step-down) do not need to be explicitly passed to the kernels, because these two cases can be distinguished by `pv.numer` (the input values for the numerators of the fractions presented in Table 3) and `pv.denom` (the value(s) for the respective denominators). If all elements of both vectors are identical, we have the step-down case. Otherwise, it is step-up.

Basically, the kernels implement the formulas of Table 3. Every step function must be evaluated at every element of either the support (for the critical values approach) or only the sorted raw p values (for the transformed p values approach). If this were done by means of `sapply` and/or `apply`, the results would be stored in a matrix, for which enough memory is reserved automatically. For a large number of hypotheses and even larger support sets, the size of this matrix would be vast and may easily take up several, if not dozens, of gigabytes of RAM. This may be too much for many workstations. As a solution to this problem, we implemented memory-conserving algorithms.

For [DBH-SD], this means, that, for each p value c.d.f. F_i, we compute the fractions $\frac{F_i(t)}{1-F_i(t)}$ for all of $\{t \in \mathcal{A} : t \geq \tau_1^{\min}\}$, with α being the significance level (see [DDR], Lemma 3) inside a `for` loop, which adds up the resulting R vectors iteratively. If the critical values are not demanded by the user, we evaluate at the observed p values instead of the support. In both cases, the number of passes of the `for` loop is identical to the number of hypotheses. For [DBH-SU], we first compute the (last) critical constant τ_m as above, but we can restrict the

computations to the set $\{t \in \mathcal{A} : t \geq \frac{\alpha}{1+\alpha}\}$ (see [DDR], Lemma 2). After that, we compute the fractions $\frac{F_i(t)}{1-F_i(\tau_m)}$ as before, but we only have to consider values of the set $\{t \in \mathcal{A} : t \leq \tau_m\}$. For the [A-DBH] procedures, the step functions are evaluated iteratively at smaller chunks of the input vectors `pv.numer` and `pv.denom`. The results of the fractions are then stored in a matrix. We found a size of 256 MiB to deliver the best performance. Depending on the number of hypotheses, m, the size and number of the chunks is determined dynamically. All p value transformations and critical constant computations are done for this submatrix. The intermediate results are then stored in vectors. This is repeated for the remaining chunks by using a `for` loop. The intermediate results are updated with each pass of the loop until all input values have been processed. The [DBR-λ] algorithm is working almost identically, but no fractions are needed and there is no step-up/step-down direction.

4.3. Run times

To illustrate the run times of `DBH`, `ADBH` and `DBR`, we used the `arabidopsisE` data set, which was once included in the **fdrDiscreteNull** package, but was removed in recent versions. From this data, a total of 17400 hypotheses along with their respective p value distributions and a vector of raw p values were derived (for more details, see Chen and Doerge (2015a)). The accumulated size of the support $\mathcal{A}$ is 1,074,398. From this data, we used subsets of the first $m = 250, 500, 1000, 3000, 5000, 7000, 10000, 17400$ hypotheses, each resulting in different support sizes, as shown in the tables in the appendix. For each subset, the median run time of 25 runs was recorded. The decision for multiple, repeated runs and their median was made in order to account for possible side loads of the workstation and to avoid overly pronounced effects of very good and especially very bad runs, so we get a robust indication of the required time. All three methods were used with the following settings:

- `alpha = 0.05`

- `direction = "sd"` and `direction = "su"` (`DBH` and `ADBH` only)

- `lambda = 0.05` (`DBR` only)

- `ret.crit.consts = TRUE` and `ret.crit.consts = FALSE`

All computations were performed with R version 3.5.1 on the following system:

- CPU: AMD Ryzen 7 1800X, 3.60 GHz

- RAM: 32 GiB DDR4, 2400 MHz

- OS: Windows 10 Education v1803

The complete results tables can be found in the appendix.

Results of critical values approach

The following plots illustrate our findings by depicting the development of the run times as a function of the product of m and the overall support size $|\mathcal{A}|$. In addition to a plot with standard axis scaling, we also employ an additional one with logarithmic axes.
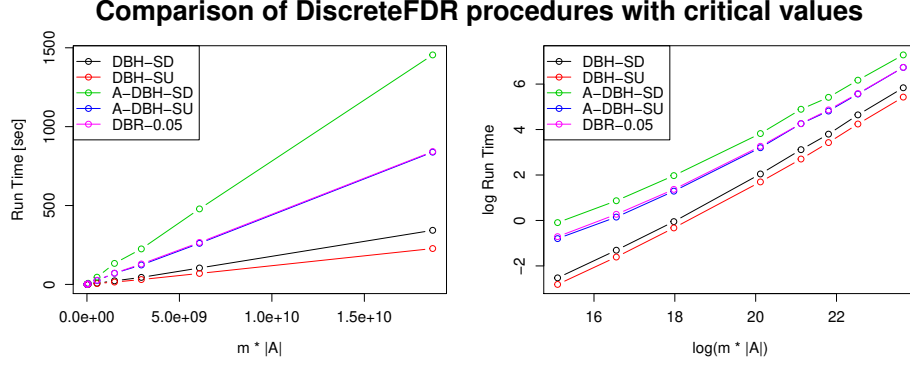
Figure 3: Run time comparison of **DiscreteFDR** procedures with computation of critical values.

From both plots, we can clearly observe that [DBH-SU] is the fastest algorithm, followed by [DBH-SD], whose computation takes about 1.5 times as long. The calculations of [A-DBH-SU] takes about 4 times and those of [A-DBH-SD] almost 7 times as long as [DBH-SU]. [DBR-0.05] needs almost exactly the same time as [A-DBH-SU], so that their respective lines in the plots overlap. In addition, the second plot shows that these proportions of run times and, as a result, the order remain stable after $m \cdot |\mathcal{A}| \approx 5,000 \cdot 300,000 = 1,500,000,000$. Furthermore, it is visible that the run times of all the procedures exhibit roughly linear growth.

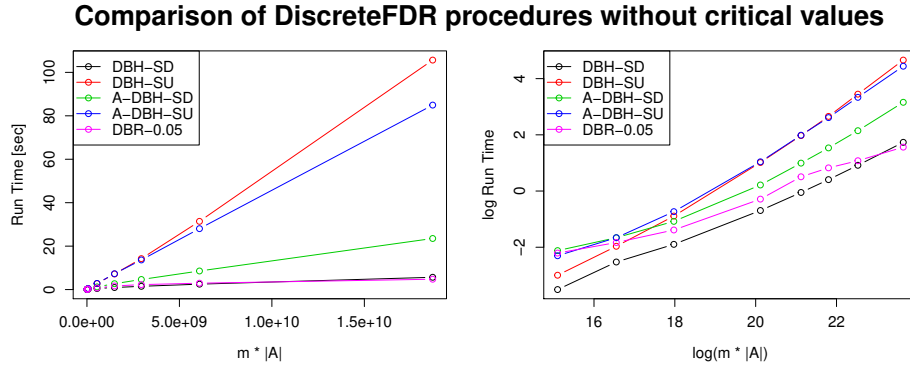*Results of transformed p values approach*

Figure 4: Run time comparison of **DiscreteFDR** procedures without computation of critical values.

Here, it is immediately apparent that the transformed p values approach is an order of magnitude faster than the ones with critical values, but recognizing a ranking is a bit more difficult. However, up to and including $m \cdot |\mathcal{A}| \approx 1,000 \cdot 64,000 = 64,000,000$, all procedures take less than a second to compute their results, which is almost unnoticeable. After that point, [DBH-SD] and [DBR-0.05] are the fastest algorithms, with [DBR-0.05] outperforming every other procedure for very large sizes. They are followed by [A-DBH-SD], [A-DBH-SU] and [DBH-SU]. The two latter ones exhibit mostly identical performance, but the higher $m \cdot \mathcal{A}$,

the higher the performance advantage of [A-DBH-SU] over [DBH-SU], although they remain the slowest methods. Their largely higher computation time is explained by the fact that these two procedures have to determine the critical value τ_m, which is responsible for 80% of the computational time, as an in-depth analysis has shown. The increasing advantage of [A-DBH-SU] over [DBH-SU] is explained by the fact that, as described before, [DBH-SU] simply adds up the fractions of evaluated c.d.f.s with a for loop, while [A-DBH-SU] uses a chunking approach, which also uses `for` loops, but requires much fewer passes than [DBH-SU]. The advantage of this approach is mitigated by the required sorting. But sill, as a result, [DBH-SU] becomes less efficient with larger sizes of $m \cdot \mathcal{A}$.

5. Further analyses

5.1. Analysis of pharmacovigilance data

To illustrate how the procedures in **DiscreteFDR** can be used for real data, we revisit the analysis of the pharmacovigilance data from Heller and Gur (2011) performed in [DDR]. This data set is obtained from a database for reporting, investigating and monitoring adverse drug reactions due to the Medicines and Healthcare products Regulatory Agency in the United Kingdom. It contains the number of reported cases of amnesia as well as the total number of adverse events reported for each of the $m = 2446$ drugs in the database. For more details we refer to Heller and Gur (2011) and to the accompanying R-package **discreteMTP** (Heller *et al.* (2012)), which also contains the data. Heller and Gur (2011) investigate the association between reports of amnesia and suspected drugs by performing for each drug a Fisher's exact test (one-sided) for testing association between the drug and amnesia while adjusting for multiplicity by using several (discrete) FDR procedures. In what follows we present code that reproduces parts of Figure 2 and Table 1 in [DDR].

We procede as in the example in section 2. Since we need to access the critical values we first determine the p values and their support for the data set `amnesia` contained for convenience in the package **DiscreteFDR**. For this, we use the option `"HG2011"` in the function `fisher.pvalues.support`.

```
R> library("DiscreteFDR")
R> data(amnesia)
R> amnesia.formatted <- fisher.pvalues.support(amnesia[, 2:3],
+                                              input = "HG2011")
R> raw.pvalues <- amnesia.formatted$raw
R> pCDFlist <- amnesia.formatted$support
```

Then we perform the FDR analysis with functions `DBH` and `ADBH` (SU and SD) and `DBR` at level $\alpha = 0.05$ including critical values.

```
R> DBH.su <- DBH(raw.pvalues, pCDFlist, ret.crit.consts = TRUE)
R> DBH.sd <- DBH(raw.pvalues, pCDFlist, direction = "sd",
+                ret.crit.consts = TRUE)
R> ADBH.su <- ADBH(raw.pvalues, pCDFlist, ret.crit.consts = TRUE)
R> ADBH.sd <- ADBH(raw.pvalues, pCDFlist, direction = "sd",
```

```
+                       ret.crit.consts = TRUE)
R> DBR.su <- DBR(raw.pvalues, pCDFlist, ret.crit.consts = TRUE)
```

By accessing the critical values we can now generate a plot similar to Figure 2 from [DDR].
Note that both [DBH-SU] and [DBH-SD] are visually indistinguishable from their adaptive
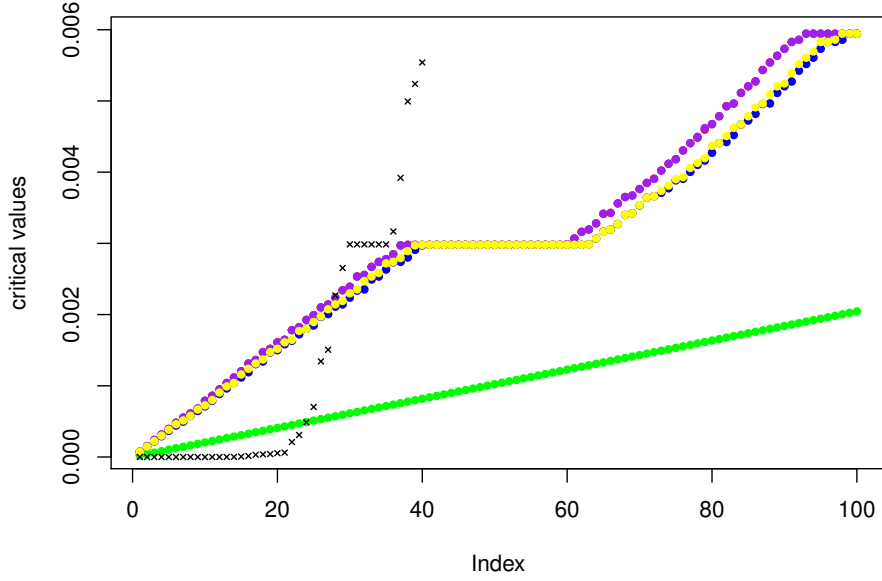counterparts.

Figure 5:   Critical values for [BH] (green dots), [DBH-SU] (orange dots), [DBH-SD] (red
dots), [A-DBH-SU] (blue dots), [A-DBH-SD] (purple dots), [DBR] (yellow dots). The sorted
raw p values are represented by asterisks.

The rejected hypotheses can be accessed via the command `$Indices`. The following code
yields some of the values from Table 1 in [DDR]:

```
R> rej.BH <- length(which(p.adjust(raw.pvalues, method = "BH") <= 0.05))
R> rej.DBH.su <- length(DBH.su$Indices)
R> rej.DBH.sd <- length(DBH.sd$Indices)
R> rej.ADBH.su <- length(ADBH.su$Indices)
R> rej.ADBH.sd <- length(ADBH.sd$Indices)
R> rej.DBR.su <- length(DBR.su$Indices)
R> c(rej.BH, rej.DBH.su, rej.DBH.sd, rej.ADBH.su, rej.ADBH.sd, rej.DBR.su)

[1] 24 27 27 27 27 27
```

The (continuous) BH rejects only 24 hypotheses whereas all the discrete procedures imple-
mented in **DiscreteFDR** are able to identify three additional drug candidates potentially
associated with amnesia.

5.2. Other types of discrete tests

In this section we sketch how **DiscreteFDR** can be used to analyse arbitrary multiple discrete tests. Jimenez-Otero *et al.* (2018) used **DiscreteFDR** to detect disorder in NGS experiments based on one-sample tests of the Poisson mean. Rather than reproducing their analysis in detail, we illustrate the general approach by using a toy example similar to the one in Section 2 and show how the test of the Poisson mean can be accomodated by **DiscreteFDR**.

To fix ideas, suppose we observe $m = 9$ independent Poisson distributed counts $N_1, \ldots, N_9$ (Jimenez-Otero *et al.* (2018) used this to model the read counts of different DNA bases). We assume that $N_i \sim \mathbf{Pois}(\lambda_i)$ and the goal is to identify cases where λ_i is larger than some pre-specified value λ_i^0, i.e., we have the (one-sided) multiple testing problem

$$H_{0i} : \lambda_i = \lambda_i^0 \qquad \text{vs.} \qquad H_{1i} : \lambda_i > \lambda_i^0.$$

As in Section 2, the goal is to adjust for multiple testing by using the [DBH-SD] procedure at FDR-level $\alpha = 5\%$. In our example the observations $n_1, \ldots, n_9$ and parameters $\lambda_1^0, \ldots, \lambda_9^0$ are given as follows:

```
      observations lambda.0
 [1,]            3      0.6
 [2,]            3      1.2
 [3,]            1      0.7
 [4,]            2      1.3
 [5,]            3      1.0
 [6,]            3      0.2
 [7,]            1      0.8
 [8,]            2      1.3
 [9,]            4      0.9
```

Denote by G_i the distribution of N_i under H_{0i} i.e., $G_i(x) = P(N_i \leq x)$. For observations $n_1, \ldots, n_9$ of $N_1, \ldots, N_9$ the p values for the above one-sided test are given by

$$p_i = P(N_i \geq n_i) = P(N_i > n_i - 1) = \overline{G_i}(n_i - 1),$$

where $\overline{G_i}(x) = P(N_i > x) = 1 - G_i(x)$ denotes the survival function of the Poisson distribution with parameter λ_i^0. Thus the raw p values are determined by the following R code:

```
R> raw.pvalues <- sapply(1:m,function(i){ppois(observations[i]-1,lambda.vector[i],
+                                      lower.tail = FALSE)})
R> raw.pvalues
```

```
[1] 0.023115288 0.120512901 0.503414696 0.373176876 0.080301397
[6] 0.001148481 0.550671036 0.373176876 0.013458721
```

Following the definition of the `qpois` function in R we define the inverse function of $\overline{G_i}$ by

$$\overline{G_i}^{-1}(p) = \min\{n \in \mathbb{N} : \overline{G_i}(n) \leq p\}$$

and obtain for the distribution function of the i-th p value under the null

$$F_i(x) = \overline{G_i}(\overline{G_i}^{-1}(x)).$$

Each function F_i is a step function with $F_i(0) = 0$, $F_i(1) = 1$ and there exists an infinite sequence of jumps at locations $1 = x_1 > x_2 > \ldots > x_n > x_{n+1} > \ldots > 0$ such that $F(x_j) = x_j$ for $j \in \mathbb{N}$.

Initially it seems that we run into a problem if we want to determine the critical values of [DBH-SD] since the supports of $F_1, \ldots, F_9$ are no longer finite (but still discrete). We can deal with this problem by using the observation from Section 3.1 that it is sufficient to consider new, restricted supports $\mathcal{A}_i \cap [s^{\min}, 1]$ where the lower threshold satisfies

$$s^{\min} \leq \tau_1^{\min} = \max\left\{t \in \mathcal{A} : t \leq y^{\min}\right\} \qquad \text{where} \qquad y^{\min} = \frac{\alpha}{m} \cdot \left(1 + \frac{\alpha}{m}\right)^{-1}. \qquad (8)$$

To determine such an $s^{\min}$ we procede as follows. Define $n_i^{\max} = \overline{G_i}^{-1}(y^{\min}) + 1$, $t_i^{\min} = \overline{G_i}(n_i^{\max} - 1)$ and set $s^{\min} = \min(t_1^{\min}, \ldots, t_9^{\min})$. It is easily checked that this choice of $s^{\min}$ satisfies (8). We can determine $s^{\min}$ by the following code

```
R> y.min <- alpha/m*(1+alpha/m)^(-1)
R> n.max <- sapply(1:m,function(w){qpois(y.min,lambda.vector[w],
+        lower.tail = FALSE)})+1
R> t.min <- sapply(1:m,function(w){ppois(n.max[w]-1,lambda.vector[w],
+                                   lower.tail = FALSE)})
R> s.min <- min(t.min)
R> s.min


[1] 0.0007855354
```

For determining the restricted supports it is actually more convenient to work with $n_i^{\max}$ than $s^{\min}$. We can subsequently use these supports as the `pCDFlist` argument in the usual way when calling the `DBH` function:

```
R> supports <- lapply(1:m,function(w){sort(ppois(0:n.max[w]-1,lambda.vector[w],
+                                   lower.tail = FALSE))})
R> DBH.sd <- DBH(raw.pvalues,supports,direction = "sd", ret.crit.consts = TRUE)
```

Figure 6 shows a summary similar to Figure 2. Applying the continuous BH procedure

```
R> p.adjust(raw.pvalues, method = "BH")


[1] 0.06934586 0.21692322 0.55067104 0.47979884 0.18067814 0.01033633
[7] 0.55067104 0.47979884 0.06056424
```

results in one rejection at FDR-level $\alpha = 5\%$, whereas the DBH step-up procedure can reject three hypotheses:

```
R> DBH.sd$Adjusted
```

```
[1] 0.039602625 0.101622881 0.580898946 0.522450788 0.101509307
[6] 0.001935955 0.626257875 0.522450788 0.033073393
```

As in Figure 2, Panel (c) presents a graphical comparison between the two procedures applied to the p values.
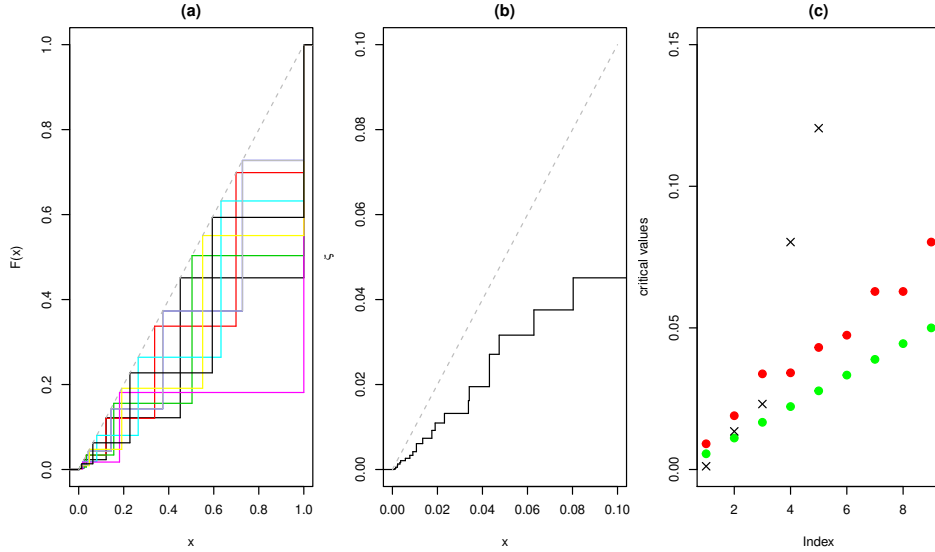

Figure 6: Panel (a) depicts the distribution functions $F_1, \ldots, F_9$ in various colours, (b) is a graph of the transformation function ξ_{SD}. The uniform distribution function is shown in light grey in (a) and (b). Panel (c) shows the [BH] critical values (green dots), the DBH step-down critical values (red dots) and the sorted raw p values (asterisks).

6. Summary and future work

Controlling the FDR for discrete tests is an important goal in many data analytic settings. In this paper, we introduced the R package **DiscreteFDR**, implementing procedures from [DDR]. These procedures come with guaranteed FDR control under independence and deal effectively with the conservativeness encountered in discrete tests.

We hope that our software will make discrete methods for FDR control more accessible to a wide audience of practitioners. More specifically, **DiscreteFDR** can be used both in an 'expert' and a 'standard' mode. For the data analyst, taking discreteness and multiplicity issues into account simultaneously may appear to be rather challenging since information on many distribution functions has to be stored, combined and evaluated. For this reason, we have included the wrapper function `fast.discrete` which applies the discrete procedures to a set of 2×2 tables, given by a matrix or data frame, where each contingency table is analysed by Fisher's exact test. Thus, this function can be seen as an implementation of a multiple Fisher test that controls FDR. For controlling the more stringent Familywise Error Rate

(FWER) for multiple exact Fisher tests, we would like to point out the R package **multfisher** which implements the approaches described in Ristl, Xi, Glimm, and Posch (2018). For those analysts who are looking for a simple to apply, off-the-shelf method, using `fast.discrete` will automatically take care of generating the list of (the support of the) distribution functions `pCDFlist`, which may otherwise be tedious work. For more expert users who want to use other tests than Fisher's exact test, the work flow, described in more detail in Section 5.2, consists of first generating the `pCDFlist` list, and then passing this on to the DBH or DBR functions.

Interfaces that generate `pCDFlist` from a given data set for a given statistical test are very helpful tools. Currently, Fisher's exact test is the only test for which our package supplies such an interface. In the future, we are planning to include helper functions similar to `fisher.pvalues.support` for further tests like the Binomial and Poisson tests.

The R package **DiscreteFDR** is available from the Comprehensive R Archive network (CRAN) at `https://cran.r-project.org/web/packages/DiscreteFDR/index.html`.

7. Acknowledgements

References

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*, **57**(1), 289–300. ISSN 00359246. URL `http://www.jstor.org/stable/2346101`.

Blanchard G, Dickhaus T, Hack N, Konietschke F, Rohmeyer K, Rosenblatt J, Scheer M, Werft W (2017). *mutoss: Unified Multiple Testing Procedures*. R package version 0.1-12, URL `https://CRAN.R-project.org/package=mutoss`.

Chen X, Doerge R (2015a). *fdrDiscreteNull: False Discovery Rate Procedure Under Discrete Null Distributions*. R package version 1.0, URL `https://CRAN.R-project.org/package=fdrDiscreteNull`.

Chen X, Doerge R (2015b). "A weighted FDR procedure under discrete and heterogeneous null distributions." *arXiv:1502.00973*.

Dickhaus T, Straßburger K, Schunk D, Morcillo-Suarez C, Illig T, Navarro A (2012). "How to analyze many contingency tables simultaneously in genetic association studies." *Statistical applications in genetics and molecular biology*, **11**(4). ISSN 1544-6115. `doi:10.1515/1544-6115.1776`. URL `http://dx.doi.org/10.1515/1544-6115.1776`.

Döhler S, Durand G, Roquain E (2018). "New FDR bounds for discrete and heterogeneous tests." *Electron. J. Statist.*, **12**(1), 1867–1900. `doi:10.1214/18-EJS1441`. URL `https://doi.org/10.1214/18-EJS1441`.

Dudoit S, van der Laan MJ (2007). *Multiple Testing Procedures and Applications to Genomics.* Springer Series in Statistics. Springer. ISBN: 978-0-387-49316-9.

Durand G, Junge F (2019). *DiscreteFDR: Multiple Testing Procedures with Adaptation for Discrete Tests.* R package version 1.2, URL https://CRAN.R-project.org/package=DiscreteFDR.

Heller R, Gur H (2011). "False discovery rate controlling procedures for discrete tests." *arXiv:1112.4627.*

Heller R, Gur H, Yaacoby S (2012). *discreteMTP: Multiple testing procedures for discrete test statistics.* R package version 0.1-2, URL https://CRAN.R-project.org/package=discreteMTP.

Heyse JF (2011). "A false discovery rate procedure for categorical data." In *Recent Advances in Bio- statistics: False Discovery Rates, Survival Analysis, and Related Topics*, pp. 43–58.

Hothorn T, Bretz F, Westfall P (2008). "Simultaneous Inference in General Parametric Models." *Biometrical Journal*, **50**(3), 346–363.

Jimenez-Otero N, de Una-Alvarez J, Pardo-Fernandez JC (2018). "Goodness-of-fit tests for disorder detection in NGS experiments." *Biometrical Journal*, **0**(0). doi:10.1002/bimj.201700284. https://onlinelibrary.wiley.com/doi/pdf/10.1002/bimj.201700284, URL https://onlinelibrary.wiley.com/doi/abs/10.1002/bimj.201700284.

Karp NA, Heller R, Yaacoby S, White JK, Benjamini Y (2016). "Improving the Identification of Phenotypic Abnormalities and Sexual Dimorphism in Mice When Studying Rare Event Categorical Characteristics." *Genetics.* ISSN 0016-6731. doi:10.1534/genetics.116.195388. http://www.genetics.org/content/early/2016/12/05/genetics.116.195388.full.pdf, URL http://www.genetics.org/content/early/2016/12/05/genetics.116.195388.

Lehmann E, Romano J (2006). *Testing Statistical Hypotheses.* Springer. ISBN 0387988645. URL http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0387988645.

Ristl R, Xi D, Glimm E, Posch M (2018). "Optimal exact tests for multiple binary endpoints." *Computational Statistics and Data Analysis*, pp. 1–17.

van den Broek E, Dijkstra MJJ, Krijgsman O, Sie D, Haan JC, Traets JJH, van de Wiel MA, Nagtegaal ID, Punt CJA, Carvalho B, Ylstra B, Abeln S, Meijer GA, Fijneman RJA (2015). "High Prevalence and Clinical Relevance of Genes Affected by Chromosomal Breaks in Colorectal Cancer." *PLOS ONE*, **10**(9), 1–14. doi:10.1371/journal.pone.0138141. URL https://doi.org/10.1371/journal.pone.0138141.

Westfall P, Wolfinger R (1997). "Multiple tests with discrete distributions." *The American Statistician*, **51**(1), 3–8. ISSN 00031305. URL http://www.jstor.org/stable/2684683.

Zhu Y, Guo W (2017). "Familywise Error Rate Controlling Procedures for Discrete Data." *ArXiv e-prints.* 1711.08147.

Zhu Y, Guo W (2017). *MHTdiscrete: Multiple Hypotheses Testing for Discrete Data.* R package version 1.0.0, URL https://CRAN.R-project.org/package=MHTdiscrete.

A. Run Time Comparison Tables

m	$|\mathcal{A}|$	Critical values	Run time
250	14442	TRUE	0.08
		FALSE	0.03
500	30873	TRUE	0.27
		FALSE	0.08
1000	64058	TRUE	0.95
		FALSE	0.15
3000	181801	TRUE	7.73
		FALSE	0.50
5000	297930	TRUE	22.51
		FALSE	0.95
7000	420162	TRUE	44.59
		FALSE	1.50
10000	608459	TRUE	104.00
		FALSE	2.50
17400	1074398	TRUE	342.88
		FALSE	5.66

Table 9: Median run times for the [DBH-SD] procedure.

m	$|\mathcal{A}|$	Critical values	Run time
250	14442	TRUE	0.06
		FALSE	0.05
500	30873	TRUE	0.20
		FALSE	0.14
1000	64058	TRUE	0.72
		FALSE	0.41
3000	181801	TRUE	5.45
		FALSE	2.76
5000	297930	TRUE	14.91
		FALSE	7.19
7000	420162	TRUE	30.76
		FALSE	14.24
10000	608459	TRUE	69.60
		FALSE	31.41
17400	1074398	TRUE	227.41
		FALSE	105.66

Table 10: Median run times for the [DBH-SU] procedure.

m	$\|\mathcal{A}\|$	Critical values	Run time
250	14442	TRUE	0.91
		FALSE	0.12
500	30873	TRUE	2.38
		FALSE	0.19
1000	64058	TRUE	7.19
		FALSE	0.34
3000	181801	TRUE	45.73
		FALSE	1.24
5000	297930	TRUE	133.17
		FALSE	2.70
7000	420162	TRUE	225.01
		FALSE	4.64
10000	608459	TRUE	478.63
		FALSE	8.56
17400	1074398	TRUE	1454.77
		FALSE	23.52

Table 11: Median run times for the [A-DBH-SD] procedure.

m	$\|\mathcal{A}\|$	Critical values	Run time
250	14442	TRUE	0.45
		FALSE	0.10
500	30873	TRUE	1.16
		FALSE	0.19
1000	64058	TRUE	3.63
		FALSE	0.48
3000	181801	TRUE	24.54
		FALSE	2.83
5000	297930	TRUE	70.62
		FALSE	7.29
7000	420162	TRUE	123.09
		FALSE	13.64
10000	608459	TRUE	260.41
		FALSE	28.05
17400	1074398	TRUE	838.53
		FALSE	84.94

Table 12: Median run times for the [A-DBH-SU] procedure.

m	$|\mathcal{A}|$	Critical values	Run time
250	14442	TRUE	0.49
		FALSE	0.11
500	30873	TRUE	1.31
		FALSE	0.16
1000	64058	TRUE	3.92
		FALSE	0.25
3000	181801	TRUE	26.10
		FALSE	0.75
5000	297930	TRUE	70.89
		FALSE	1.66
7000	420162	TRUE	129.52
		FALSE	2.28
10000	608459	TRUE	266.00
		FALSE	2.94
17400	1074398	TRUE	842.2
		FALSE	4.77

Table 13: Median run times for the [DBR] procedure at $\lambda = 0.05$.

Affiliation:

Guillermo Durand
Sorbonne Université,
Laboratoire de Probabilités, Statistique et Modélisation
4, place Jussieu
75005 Paris, France
E-mail: Guillermo.Durand@upmc.fr

Florian Junge
Darmstadt University of Applied Sciences
Darmstadt Institute for Statistics and Operations Research
Schöfferstraße 3
64295 Darmstadt, Germany
E-mail: florian.junge@h-da.de

Sebastian Döhler
Darmstadt University of Applied Sciences
Faculty of Mathematics and Natural Sciences
Darmstadt Institute for Statistics and Operations Research
Schöfferstraße 3
64295 Darmstadt, Germany
E-mail: sebastian.doehler@h-da.de

Etienne Roquain
Sorbonne Université,
Laboratoire de Probabilités, Statistique et Modélisation
4, place Jussieu
75005 Paris, France
E-mail: `Etienne.Roquain@upmc.fr`