Proceedings of Machine Learning Research vol 145:1–32, 2021

# Dynamic Algorithms for Online Multiple Testing

**Ziyu Xu** XZY@CMU.EDU
**Aaditya Ramdas** ARAMDAS@CMU.EDU
*Department of Statistics and Data Science, Machine Learning Department*
*Carnegie Mellon University, Pittsburgh, PA 15213.*

## Abstract

We derive new algorithms for online multiple testing that provably control false discovery exceedance (FDX) while achieving orders of magnitude more power than previous methods. This statistical advance is enabled by the development of new algorithmic ideas: earlier algorithms are more "static" while our new ones allow for the dynamical adjustment of testing levels based on the amount of wealth the algorithm has accumulated. We demonstrate that our algorithms achieve higher power in a variety of synthetic experiments. We also prove that SupLORD can provide error control for both FDR and FDX, and controls FDR at stopping times. Stopping times are particularly important as they permit the experimenter to end the experiment arbitrarily early while maintaining desired control of the FDR. SupLORD is the first non-trivial algorithm, to our knowledge, that can control FDR at stopping times in the online setting.

**Keywords:** Online multiple testing, false discovery exceedance, false discovery rate, dynamic schedules, stopping times.

## 1. Introduction to Online Multiple Testing

Online multiple hypothesis testing has become an area of recent interest due to the massively increasing rate at which analysts are able to test hypotheses, both in the sciences and in the tech industry. This has been a result of the growth in computational resources for automating the generation and testing of new hypotheses (and collection of data to test them).

Online multiple testing is an abstraction of the scientific endeavor — the broad goal is to test a (potentially infinite) sequence of hypotheses $H_1$,$H_2$,... one by one over time. For example $H_i$ could state that *Alzheimer's drug $i$ is no more effective than a placebo*, and if we reject this hypothesis, we are effectively proclaiming a "discovery". Some hypotheses are truly "null", meaning that there really is no effect of interest, and the rest are "non-null", but we do not know which are which, and we must use the data collected to identify as many non-null hypotheses as possible (making "true discoveries"), while not rejecting too many truly null hypotheses (avoid making "false discoveries"). Importantly, in the online setting, a decision to reject (or not reject) hypothesis $H_k$ at time $k$ is irrevocable and cannot be altered at a future time.

It is important to note that the setup assumes that algorithm (or analyst) never finds out whether a hypothesis was correctly rejected or not. In other words, we do not know (until perhaps years later, after a lot of followup time and money has been invested) which of our proclaimed discoveries were true discoveries and which were false. Thus the false discovery proportion (FDP), which is the ratio of false to total discoveries is an unobserved random variable, and we would like to make many discoveries (have high "power") while keeping the FDP small.

Previous algorithms for multiple testing have primarily focused on the false discovery rate (FDR), which is the expectation of the FDP. However, in practice, users may be more interested in a probabilistic bound of the tail of the FDP as it offers a stricter guarantee about the distribution of the FDP. Thus, we are interested in controlling the false discovery exceedance (FDX) , which is the probability the FDP exceeds a predefined threshold. To accomplish this, we first must describe how hypotheses are tested in the online multiple testing problem. As stated prior, we are testing a infinite sequence of hypotheses $H_1,H_2,...$, and each hypothesis is either null or non-null. For each hypothesis, we perform some type of experiment, analyze the data, and arrive at a summary statistic known as a "p-variable" that is supported only in $[0,1]$. Colloquially, p-variables are known as p-values, but we use the name "p-variable" to emphasize the fact that p-variables are random variables, and not a probability value. We use "p-value" to denote the specific value a p-variable is sampled at. Typically, p-variables are the result of making certain sampling assumptions about the data, and deriving some type of statistic with a known distribution from the data — an excellent reference on how these statistics are derived is chapter 10 in Wasserman (2004). Hence, $P_k$ is the p-variable whose distribution is based on the data drawn under hypothesis $H_k$, and $p_k$ will be the corresponding p-value to $P_k$. If $H_k$ is null, then the the distribution of $P_k$ must be stochastically larger than uniform. We will formalize this notion later, but the key message here is that we essentially know what the distribution of the p-variable under the null hypothesis is.

As a result, an online multiple testing algorithm is primarily concerned with the stream of p-values $p_1,p_2,...$, which correspond to the sequence of hypotheses being tested. For each hypothesis $H_k$, the algorithm must output an alpha value, $\alpha_k$, in $[0,1]$, based solely on the previous p-values it has received i.e. $p_1,...,p_{k-1}$. Critically, it must output this alpha value before receiving the next p-value $p_k$. $H_k$ is then rejected if $p_k$ is less than or equal to $\alpha_k$. We denote the set of rejected hypotheses up to the $k$th hypothesis as $\mathcal{R}_k \subseteq \{1,...,k\}$. The set of null hypothesis is denoted as $\mathcal{H}_0 \subseteq \mathbb{N}$. We let the set of null rejections (i.e. false discoveries) in the first be $k$ hypotheses be denoted as $V_k = \mathcal{H}_0 \cap \mathcal{R}_k$. Thus, the FDP of $\mathcal{R}_k$ is defined as follows:

$$\mathrm{FDP}(\mathcal{R}_k) \equiv \frac{|V_k|}{|\mathcal{R}_k| \vee 1}.$$

Define the (simultaneous, or time-uniform) $\mathrm{FDX}_K$ as the probability that the FDP ever exceeds a threshold $\epsilon^* \in (0,1)$ after time $K$:

$$\mathrm{FDX}_K^{\epsilon^*} \equiv \mathbf{Pr}\left(\sup_{k \geq K} \mathrm{FDP}(\mathcal{R}_k) \geq \epsilon^*\right).$$

The FDR in the online setting is a pointwise guarantee for each hypothesis i.e. it is the largest expected FDP for any fixed time after $K$:

$$\mathrm{FDR}(\mathcal{R}_k) \equiv \mathbb{E}[\mathrm{FDP}(\mathcal{R}_k)], \qquad \mathrm{FDR}_K \equiv \sup_{k \geq K} \mathrm{FDR}(\mathcal{R}_k).$$

(In prior work FDR and FDX have been defined with $K = 1$.) We also provide the first algorithm that controls a related time-uniform error metric, which is the expectation of the largest FDP:

$$\mathrm{SupFDR}_K \equiv \mathbb{E}\left[\sup_{k \geq K} \mathrm{FDP}(\mathcal{R}_k)\right].$$

Any algorithm that controls $\mathrm{SupFDR}_1$ at a level $\ell$ will also control $\mathrm{FDR}_1$ at $\ell$ as well, since SupFDR is strictly larger than FDR. One central difference is that $\mathrm{FDR}_1 \leq \ell$ does not yield a guarantee at a data-dependent stopping time $\tau$, but $\mathrm{SupFDR}_1 \leq \ell$ does imply that $\mathrm{FDR}(\mathcal{R}_\tau) \leq \ell$. This yields the first FDR guarantee at stopping times in the literature. In past work "anytime FDR control" really held at any fixed time (when the time is specified in advance), but controlling SupFDR truly makes the aforementioned phrase actionable in the online setting, since it is a simultaneous guarantee over all time. We expand on the relationship between these two error metrics in Section 1.2. We aim to maximize *power*, which is the expected proportion of non-null hypotheses that are rejected:

$$\mathrm{Power}(\mathcal{R}_k) \equiv \mathbb{E}\left[\frac{|\mathcal{R}_k \cap \mathcal{H}_0^c|}{|\mathcal{H}_0^c|}\right].$$

Thus, the problem of online multiple testing is to design an algorithm that rejects as as many non-null hypotheses as possible while not rejecting null hypotheses to a degree that is sufficient to satisfy the desired level of control on an error metric (e.g. FDR, FDX, SupFDR).

> We do not make guarantees about the optimality of power. This is because we do not have any information about the alternate hypothesis and make no assumptions about the distribution and frequency of non-null hypotheses. However, we can assert that the error metric is provably controlled, since it is only dependent on null hypotheses.

The LORD algorithm, introduced by Javanmard and Montanari (2018), is an online multiple testing algorithm that provably controls the FDR at level $\ell$ (i.e. guarantees FDR will be less than $\ell$). We will refer to Javanmard and Montanari (2018) as JM in the sequel. JM show that LORD, with additional constraints, controls FDX. We will refer to this version of LORD as LORDFDX. LORDFDX *is the only existing rule that controls* FDX *at a set level to the best of our knowledge.* We determine, through experiments, that LORDFDX is extremely conservative and we design an algorithm that significantly improves power while controlling FDX at the same level.

Our proofs employ the post-hoc probabilistic bounds on the FDP developed by Katsevich and Ramdas (2020), which we refer to as KR henceforth. KR do not propose an algorithm for guaranteeing a certain level of FDX control, but rather formulate an algorithm-agnostic bound based on *any* sequence of alpha values $\{\alpha_i\}_{i \leq k}$ and rejection sets $\mathcal{R}_k$, in order to form a time-uniform confidence interval for the FDP. We view our work as an algorithmic mechanization of these bounds such that the FDX can be controlled by the practitioner in a premeditated fashion.

### 1.1. Notation and P-Variable Assumptions

**Abbreviations.** We abbreviate a few works that will be referenced often. As mentioned in the prequel, JM refers to Javanmard and Montanari (2018) and KR refers to Katsevich and Ramdas (2020). FS refers to Foster and Stine (2008), which is the original work in this area.

**Notation.** Recall that $P_k$ is the $k$th p-variable and $\alpha_k$ is the $k$th alpha value. Let $R_k = \mathbf{I}\{P_k \leq \alpha_k\}$ indicate whether the $k$th hypothesis is rejected. Generally, script variables such as $\mathcal{H}, \mathcal{R}, \mathcal{F}$ are sets. Capital letters, such as $R, V, W, P$ are random variables. Greek letters, such as $\alpha, \beta, \gamma, \delta, \epsilon$, represent user selected parameters for the algorithm. We collect the "past rejection information" into a filtration:

$$\mathcal{F}_k \equiv \sigma(\{R_i\}_{i \leq k}), \tag{1}$$

with $\mathcal{F}_0$ being the trivial $\sigma$-algebra. Note that $\{\alpha_k\}_{k \in \mathbb{N}}$ is predictable with respect to $\{\mathcal{F}_k\}_{k \in \mathbb{N}}$ meaning that $\alpha_k$ is $\mathcal{F}_{k-1}$-measurable.

**P-variable assumptions.** In classical hypothesis testing, a p-variable is a random variable whose distribution is stochastically larger than uniform under the null, meaning that if the null hypothesis is true, we would have $\mathbf{Pr}(P \leq s) \leq s$ for all $s \in [0,1]$. For the online setting, the weakest possible assumption one could make is that conditional on the past, the p-variables still satisfy such a condition. This is usually formalized (by FS, Ramdas et al. 2017, and KR, for example), by the following *conditional superuniformity* assumption about null p-variables:

$$\mathbf{Pr}(P_{k+1} \leq s \,|\, \mathcal{F}_k) \leq s \text{ for } k \in \mathcal{H}_0,\, s \in [0,1]. \tag{2}$$

Note that the above assumption is weaker than assuming that the p-variables are independently, superuniformly distributed under the null hypothesis:

$$\mathbf{Pr}(P_k \leq s) \leq s \text{ for } k \in \mathcal{H}_0,\, s \in [0,1] \text{ and } \{P_k\}_{k \in \mathcal{H}_0} \text{ are independent}, \tag{3}$$

which is also commonly made in the online FDR literature (in JM, for example). For the rest of the paper, we make no further assumptions on the relationship between on the hypotheses, $H_1, H_2, \ldots$ other than (2). We include assumption (3) because it implies (2) and has been used to prove error control in prior work e.g. LORD requires this assumption for FDR control. Hence, the online multiple testing is a relatively assumption-light framework that can be adapted to many practical settings. Lastly, we define $t_r$ to be the time step when the $r$th rejection is made, where $r$ is a positive integer.

### 1.2. Summary of Contributions

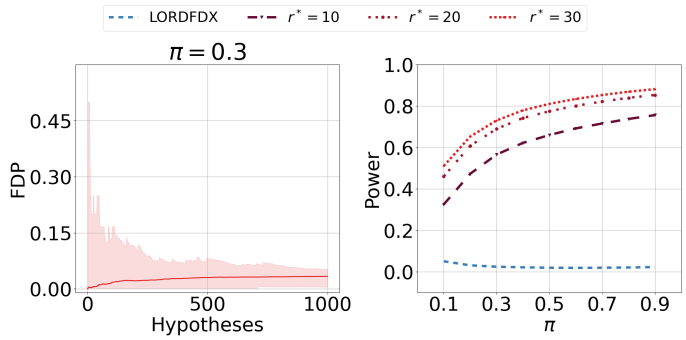



Figure 1: The right plot is the power of SupLORD, for different delays in rejections, $r^*$, vs. LORDFDX from JM. The left plot is the average FDP of SupLORD with $r^* = 30$ across hypotheses tested for non-null likelihoods $\pi \in \{0.1, 0.3\}$. All algorithms control FDX for $\epsilon^* = 0.15$ at a level of $\delta^* = 0.05$. Experiment details are in Section 3.2. The shaded area is a $1-\delta^*$ confidence band for the FDP — it is smaller than $\epsilon^*$ for the majority of hypotheses. This shows FDX control at $\epsilon^*$ holds for a substantial portion of the hypotheses. Also, the power of SupLORD increases with $r^*$. Thus, SupLORD has significant FDX control and more power than LORDFDX (by JM).

**Delayed FDX control.** We provide a formula for SupLORD, which controls FDX at a set level, and has more power than LORDFDX (by JM). In fact, we observe empirically that LORDFDX rejects only a few hypotheses before becoming unable to reject any more hypotheses, and performs no better than the baseline approach we discuss in Section 2.1. We prove SupLORD controls FDX by using a time-uniform confidence interval introduced in KR on the FDP. A novel feature of our algorithm is that we may choose the number of rejections after which we begin controlling FDX in

exchange for more power. Practically, users often test a large number of hypotheses, and make many discoveries. Thus, the small number of rejections with no FDX control is an insignificant fraction of the experiment. The right plot in Figure 1 shows that having the FDX bound apply after only 10 rejections is sufficient for a noticeable increase in power over LORDFDX. The left plot indicates that the FDX is still controlled for a substantial proportion of the hypotheses, despite delaying control for 30 rejections. Consequently, we show through experimental results[1] that by affording this flexibility to our algorithm, it can improve power with no practical deficit.

**Dynamic scheduling.** Many multiple testing algorithms, including SupLORD, are based upon a notion of generalized alpha-investing introduced by Aharoni and Rosset (2014) which maintains a nonnegative value known as wealth. Wealth quantifies the error budget of the algorithm, and is an upper bound on the size of the next alpha value. For our error metrics of interest, an algorithm gains wealth by making rejections. We discuss and formalize the role of wealth in Section 2. Deciding how to allocate an algorithm's wealth to alpha values is a key challenge of designing online multiple testing algorithms. We devise a method for formulating alpha values that *dynamically* chooses larger alpha values when wealth is large. Through experiments, we demonstrate that SupLORD with dynamic scheduling outperforms spending schedules introduced in JM, and fully utilizes its wealth and increases its power as a result.

**Theoretical improvements in FDR control.** The SupLORD algorithm is able to control not only FDX, but also FDR and SupFDR at a set level of control. Table 1 illustrates the difference in assumptions and error control guarantees between SupLORD and existing methods. Without imposing more restrictive independence assumptions and reducing the regime of control to fixed times, previous algorithms (FS, JM, Ramdas et al. 2017, 2018; Tian and Ramdas 2019) could only control a modified FDR:

$$\mathrm{mFDR}(\mathcal{R}_k) \equiv \frac{\mathbb{E}[|V_k|]}{\mathbb{E}[|\mathcal{R}_k| \vee 1]}. \tag{4}$$

By controlling SupFDR, SupLORD controls FDR under the same guarantees and assumptions as previous algorithms that controlled mFDR. Specifically, SupLORD offers the following theoretical improvements for control of FDR:

1. *Control at stopping times.* Stopping times are random times that are a function of past events. Previous results have only controlled FDR at fixed times. Consequently, this enables to the user to stop SupLORD prematurely, and still maintain valid FDR bounds.

2. *Non-monotonicity.* The spending schedule in prior work required $\alpha_k$ to be **monotone**, that is:

   $$\alpha_k \text{ is coordinatewise nondecreasing w.r.t. past rejections } R_1,\ldots,R_{k-1} \text{ for all } k \geq 1. \tag{5}$$

   However, SupLORD has a provable FDR guarantee without this constraint on its spending schedule due to using techniques from KR. Thus, when using non-monotonic schedules such as dynamic scheduling, SupLORD maintains a provable FDR guarantee.

3. *Non-independence*. Prior algorithms require independent p-values to have FDR control. SupLORD only requires a natural, weaker, baseline assumption we formalize in (2). This assumption is required to prove any known guarantee for online multiple testing algorithms.

1. Code for the figures and experiments in this paper can be found at https://github.com/neilzxu/suplord.

Table 1: Comparison of the error metrics controlled under different assumptions of online multiple testing algorithms discussed in Section 1.2. Note that assumption (3) implies assumption (2), and hence any guarantees achieved under (2) are also achieved under (3). To achieve FDR control, all algorithms except SupLORD require both independent superuniform p-variables, and monotone spending schedules — SupLORD can offer FDR guarantees under only conditional superuniformity (2). In addition, SupLORD is the only algorithm that offers control of FDR at stopping times and FDX control without alpha-death, as LORDFDX suffers from alpha-death.

| **Method** | **Guarantees** | |
|---|---|---|
| LORD, SAFFRON, ADDIS (JM, Ramdas et al. 2017 Ramdas et al. 2018, Tian and Ramdas 2019) | *If p-variables satisfy conditional superuniformity* (2)*:*<br>• mFDR at stopping times | *If p-variables satisfy independence, superuniformity* (3)*, and monotone spending* (5)*:*<br>• FDR at fixed times<br>• mFDR at stopping times |
| SupLORD | *If p-variables satisfy conditional superuniformity* (2)*:*<br>• FDR at stopping times (Corollary 5)<br>• FDX w/o alpha-death (Theorem 2)<br>• mFDR at stopping times (Theorem 10) | |

## 2. Generalized Alpha-Investing (GAI)

One of the key paradigms in online hypothesis testing is notion of alpha-investing that was first introduced by Foster and Stine (2008), and expanded upon in later works (Aharoni and Rosset, 2014; Ramdas et al., 2017). Our primary error metric of interest in this section will be FDR, since prior algorithms, including LORD, have focused on controlling FDR. However, this paradigm is generalizable to any of the error metrics we consider in this paper. In particular, we will see how we use this idea to derive our new algorithm, SupLORD, in Section 3.

### 2.1. Alpha-Spending and Alpha-Death

To elucidate alpha-investing, we first discuss the notion of alpha-spending, which is an Bonferroni correction. We control the FDR (indeed the familywise error rate, a more stringent criterion) by ensuring that the following constraint is satisfied at every $k$:

$$\sum_{i=1}^{k} \alpha_k \leq \ell. \tag{6}$$

In condition (6), we can view $\ell$ as our error budget, or **wealth**, since the sum of all alpha values adds up to $\ell$. Thus, we can imagine this as starting out with wealth of $\ell$, and spending no more than our current wealth on the current alpha value for each hypothesis. This method for choosing alpha values is known as alpha-spending. We can explicitly define the wealth at time $k$ for alpha-spending to be:

$$W(k) \equiv \ell - \sum_{i=1}^{k} \alpha_i. \tag{7}$$

Definition (7) indicates that wealth is monotonically decreasing when alpha-spending. As more hypotheses are tested, the alpha values will shrink towards zero. Thus, the algorithm will fail to reject nearly every future hypothesis after enough hypotheses are tested. Ramdas et al. (2017) term the shrinkage or disappearance of alpha values and consequent failure to reject hypotheses after a point in time as **alpha-death**. Alpha-spending suffers from alpha-death because the bound is extremely conservative — Bonferroni controls the probability of making even a single false rejection to be at level $\ell$ as opposed to simply controlling the FDR to be at $\ell$.

### 2.2. Generalized Alpha-Investing Algorithms

**An alpha-investing example: LORD.** To avoid alpha-death, we can choose a less conservative bound than Bonferroni while maintaining FDR control. Ramdas et al. (2017) provide an estimator that upper bounds the FDR:

$$\widehat{\mathrm{FDP}}_{\mathrm{LORD}}(\mathcal{R}_k) = \frac{\sum_{i=1}^{k} \alpha_i}{|\mathcal{R}_k| \vee 1}. \tag{8}$$

Notably, this is also the estimator that LORD uses to control FDR. Thus, we simply need to control $\widehat{\mathrm{FDP}}_{\mathrm{LORD}}$ at level $\ell$, for the FDR to be controlled at the same level. Hence, we get a different wealth formulation than in the alpha-spending case:

$$W(k) \equiv \ell(|\mathcal{R}_k| \vee 1) - \sum_{i=1}^{k} \alpha_i. \tag{9}$$

The key difference between this formulation and alpha-spending is that wealth can be earned. Thus, alpha-death is avoided since $\alpha_i$ need not shrink to zero. We obtain a reward of $\ell$ each time the algorithm makes a rejection[2] i.e. increases the cardinality of $\mathcal{R}_k$. The wealth reward follows from (8) — when more hypotheses are rejected, the denominator in (8) increases, which allows for a larger numerator while maintaining the same $\widehat{\mathrm{FDP}}_{\mathrm{LORD}}$ value. Thus, an algorithm may carefully "invest" its wealth on alpha values in a way that maximizes the number of rejections.

**Generalized alpha-investing (GAI).** GAI algorithms have the following wealth update:

$$W(k) \equiv \beta_0 + \sum_{i=1}^{k} \beta_i R_i - \sum_{i=1}^{k} \alpha_i, \tag{10}$$

where $\{\beta_i\}_{i \in \mathbb{N} \cup \{0\}}$ is the boost sequence and is nonnegative. $\beta_i$ is the boost in wealth that is earned from the algorithm making the rejection on the $i$th hypothesis. Similar to alpha values, we require $\{\beta_k\}_{k \in \mathbb{N}}$ to be predictable with respect to $\{\mathcal{F}_k\}_{k \in \mathbb{N}}$ and $\beta_0$ to be a constant. We can recover (7) by setting $\beta_i = 0$, and we can recover (9) by setting $\beta_i = \ell$. The key invariant that all alpha-investing algorithms maintain is the following:

$$\alpha_k \leq W(k-1). \tag{11}$$

Thus, selection of alpha values respects this notion of wealth and ensures it is always nonnegative.

2. In this formulation, the initial wealth is $\ell$ and no wealth is gained from the first rejection, but this can be easily amended so that the sum of the initial wealth and first reward equals $\ell$ using any other split.

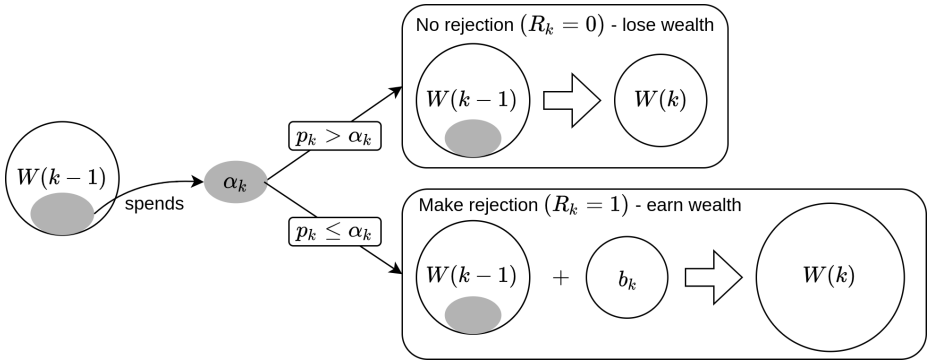



Figure 2: Diagram of the alpha-investing paradigm. The algorithm spends wealth at each hypothesis $k$ and earns wealth if it makes a rejection, and loses $\alpha_k$ wealth if it does not.

**Spending schedule.** JM also proposed two methods of choosing alpha values (i.e. spending schedules) based on wealth. Let $\{\gamma_i\}_{i\in\mathbb{N}}$ be a **spending sequence** that has the following properties:

$$\gamma_i \geq 0 \text{ for all } i \in \mathbb{N} \tag{12a}$$

$$\sum_{i=1}^{\infty} \gamma_i = 1. \tag{12b}$$

JM propose the following spending schedules:

$$\alpha_k = \begin{cases} \gamma_k \beta_0 + \sum_{j=1}^{|\mathcal{R}_{k-1}|} \gamma_{k-t_j} \beta_{t_j} & \text{STEADY} \\ \gamma_{k-t_{|\mathcal{R}_{k-1}|}} W(t_{|\mathcal{R}_{k-1}|}) & \text{AGGRESSIVE.} \end{cases} \tag{13}$$

While both are valid spending schedules in the sense that neither would violate condition (11), only STEADY can be used to provide provable FDR control in LORD, since STEADY is the only monotone rule (5). Intuitively, this means that when more rejections are made, a monotone spending schedule should output larger alpha values at all future time steps than it would if fewer rejections were made. JM show that STEADY satisfies monotonicity while AGGRESSIVE does not. Hence, AGGRESSIVE does not provide any FDR guarantees when used with LORD. In Section 5, we will discuss how SupLORD can maintain FDR control even when using non-monotone spending schedules.

For the spending sequence, a default choice is to set $\gamma_i \propto \frac{\log(i \vee 2)}{i \exp(\sqrt{\log i})}$, which is motivated by optimizing a lower bound for the power of LORD in the Gaussian setting in JM.

## 3. SupLORD: Delayed FDX Control

We propose a new alpha-investing algorithm, SupLORD, which provides control over the FDX. SupLORD requires the user to specify $r^*$, a threshold of rejections after which the error control begins to apply, $\epsilon^*$, the upper bound on the FDP, and $\delta^*$, the probability at which the FDP exceeds $\epsilon^*$ at any time step after making $r^*$ rejections. We derive this algorithm from a new estimator, $\overline{\text{FDP}}$, that gives probabilistic, time-uniform control over the FDP. Notably, SupLORD requires no assumptions about relationships between hypotheses tested except for condition (2) on p-variables of null hypotheses.

### 3.1. Deriving the Boost Sequence

To prepare for the $\overline{\text{FDP}}$, define the following quantities, respectively:

$$\widehat{V}_k \equiv \sum_{i=1}^{k} \alpha_i \,, \qquad \widehat{\overline{\text{FDP}}}(\mathcal{R}_k) \equiv \frac{\widehat{V}_k + 1}{|\mathcal{R}_k|}.$$

Now, for any $\delta \in (0,1)$, define

$$\overline{\log}\left(\frac{1}{\delta}\right) \equiv \frac{\log(\frac{1}{\delta})}{\log\big(1+\log(\frac{1}{\delta})\big)}.$$

Hence our estimator of interest is defined as follows:

$$\overline{\text{FDP}}(\mathcal{R}_k) \equiv \overline{\log}\left(\frac{1}{\delta^*}\right) \cdot \widehat{\overline{\text{FDP}}}(\mathcal{R}_k).$$

**Fact 1 (Theorem 4 from KR)** *Let $\{\alpha_k\}_{k\in\mathbb{N}}$ be any sequence of alpha values predictable with respect to filtration $\mathcal{F}_k$ in definition (1). Assuming conditional superuniformity (2), the following uniform bound holds:*

$$\mathbf{Pr}\big(\text{FDP}(\mathcal{R}_k) \le \overline{\text{FDP}}(\mathcal{R}_k) \text{ for all } k \ge 1\big) \ge 1-\delta.$$

Thus, to control FDX, SupLORD must ensure $\overline{\text{FDP}}(\mathcal{R}_k) \le \epsilon^*$ for all $k \ge t_{r^*}$. This is equivalent to requiring the following two conditions on $\{\beta_j\}_{j\in\mathbb{N}\cup\{0\}}$:

$$\sum_{i=0}^{t_{r^*-1}} \beta_i R_i \le \frac{\epsilon^* r^*}{\overline{\log}(\frac{1}{\delta^*})} - 1 \text{ for } i \le t_{r^*-1} \tag{14a}$$

$$\beta_i \le \frac{\epsilon^*}{\overline{\log}(\frac{1}{\delta^*})} \text{ for all } i > t_{r^*-1}. \tag{14b}$$

Unlike previously discussed alpha-investing algorithms, SupLORD has two different conditions for its boost sequence. This is the result of SupLORD's capability to delay control of FDX up to $r^*$ rejections. We can think of SupLORD as operating in two phases. The first phase of SupLORD is when it has not made $r^*-1$ rejections yet. The constraint on this phase is formulated in condition (14a). In this phase, we are not concerned with how much boost in wealth the algorithm receives at each rejection in the first $r^*-1$ rejections. Thus, we only restrict the *total wealth* SupLORD attains in the first $r^*-1$ rejections so that it does not attain so much wealth that it can violate the bound on $\overline{\text{FDP}}$ when the $r^*$th rejection occurs.

The second phase, when we have made $r^*$ rejections, is an analog to the alpha-investing schemes of LORD and alpha-spending, except for the estimator $\overline{\text{FDP}}$. Condition (14b) restricts the wealth boost for each rejection to be limited so that $\overline{\text{FDP}}$ is controlled to be less than $\epsilon^*$. As a result, we can prove the following lemma:

**Lemma 1** *Any GAI algorithm with a boost sequence $\{\beta_i\}_{i\in\mathbb{N}\cup\{0\}}$ that satisfies conditions (14) will ensure $\overline{\text{FDP}}(\mathcal{R}_k) \le \epsilon^*$ for all times $k \ge t_{r^*}$ such that a rejection is made at time $k$.*

A formal proof this statement is shown in Appendix A.1. Thus, by combining Lemma 1 with Fact 1, we can derive the following guarantee for SupLORD.

**Theorem 2** *Assuming conditional superuniformity (2), let $0<\epsilon^*,\delta^*<1$, $r^*\geq 1$ be a positive integer, and $t_{r^*}$ be a random variable that is the time when the $r^*$th rejection is made. A GAI algorithm with a boost sequence $\{\beta_i\}_{i\in\mathbb{N}\cup\{0\}}$ that satisfies conditions (14) guarantees:*

$$\mathrm{FDX}_{t_r^*}^{\epsilon^*}\leq\delta^*.$$

A default choice for selecting a boost sequence that satisfies conditions (14) is as follows:

$$\beta_i=\begin{cases}\dfrac{\frac{\epsilon^* r^*}{\log\left(\frac{1}{\delta^*}\right)}-1}{r^*} & i\leq t_{r^*-1},\\ \dfrac{\epsilon^*}{\log\left(\frac{1}{\delta^*}\right)} & i>t_{r^*-1}.\end{cases}\tag{15}$$

This boost sequence is tight with both upper bounds in conditions (14), and it evenly distributes the upper bound in condition (14a) across the initial wealth and first $r^*-1$ rejections.

### 3.2. Numerical Simulations and Real Data Experiments for FDX Control

We primarily explore the power of our procedures through simulations. The reason for this is simple: when working with real data, we do not know which of our discoveries are true and which are false (otherwise, scientific hypothesis testing would be very easy; we would simply avoid the false ones). Definitively determining which discoveries are true in real data is generally intractable and requires extensive effort (e.g., after a large-scale genome-wide association study, if we proclaim that we have discovered a gene that regulates our propensity for some disease, it may take many years of followup targeted 'gene knockout' experiments to determine if our discovery was genuinely true, or a false alarm). In contrast, with synthetic hypotheses and data, we know which hypotheses are null by construction, and can evaluate the power of our algorithms and confirm they do satisfy control of the desired error metric. Nevertheless, we do compare our methods on a real world dataset, the International Mouse Phenotype Consortium (IMPC) dataset that was created by Karp et al. (2017) and has been used to test other online multiple testing algorithms in prior work (Tian and Ramdas, 2021). However, because of the aforementioned issues, we can only count the total number of discoveries as a proxy for each algorithm's power (proportional to the total number of *true* discoveries).

We perform experiments for several baseline methods that control false discoveries.

1. **LORD and LORDFDX.** We use the version of LORD and LORDFDX with a STEADY spending schedule from Ramdas et al. (2017) that is more powerful than its original formulation in JM, and provably controls FDR to be under level $\ell$.

2. **Alpha-spending.** This is simply the online Bonferroni rule that controls the probability of any false discovery to be under level $\ell$. We also use the STEADY schedule for spending.

We compare SupLORD against existing methods for false discovery control. We choose $\delta^*=0.05$, $\epsilon^*=0.15$ for SupLORD and LORDFDX. In SupLORD, we choose $r^*=30$ for our simulations and $r^*=100$ for the IMPC dataset, since it contains many more hypotheses than our simulations — we also use the default boost sequence in (15). We set $\ell=0.05$ for LORD and alpha-spending. For LORD and LORDFDX, we choose $\beta_0=0.1\ell$, $\beta_1=0.9\ell$ and $\beta_i=\ell$ for all $i>1$. All methods use the default $\{\gamma_i\}_{i\in\mathbb{N}}$ where $\gamma_i\propto\frac{\log(i\vee 2)}{i\exp(\sqrt{\log i})}$ as stated in Section 2.

**Synthetic generation of p-values.** Our simulations follow a similar setup to Ramdas et al. (2018). We compute p-values based on a one sided test for Gaussian values i.e. we sample a Gaussian $Z_i$ and compute one-sided p-values for whether $\mu_i > 0$ using the statistic $P_i = \Phi(-Z_i)$ where $\Phi$ is the standard Gaussian c.d.f. Let $\pi_i$ be the probability of the $i$th hypothesis being non-null. We generate our $Z_i$ as follows:

$$B_i \sim \text{Bern}(\pi_i), \qquad Z_i \sim \begin{cases} \mathcal{N}(0,1) & B_i = 0 \\ \mathcal{N}(\mu_i,1) & B_i = 1. \end{cases}$$

Here $B_i$ is a Bernoulli variable that denotes whether the $i$th hypothesis is null. If $B_i = 0$, then we sample $Z_i$ from the null distribution, which is the standard Gaussian. If $B_i = 1$, then we draw $Z_i$ from the non-null distribution, which is a shifted standard Gaussian with mean $\mu_i$. We may select different signal strengths $\mu_i$, and non-null likelihoods $\pi_i$ at each time step for our experiments. For our primary simulation, we choose constant values $\mu,\pi$ such that $\mu_i = \mu, \pi_i = \pi$ across all $i$. Addition simulations with other ways of selecting $\mu_i$ and $\pi_i$ are discussed in Appendix E.

Ultimately, we generate 200 experiments with 1000 hypotheses for each data setting. Standard errors are negligible in all of our numerical results, and we do not plot them.

**Real data.** The IMPC dataset consists of a series of experiments that aims to identify which mouse phenotypes each mouse protein coding gene has an effect on. Karp et al. (2017) analyze the raw experimental data, and produce a series of p-values corresponding to each pairwise combination of gene and phenotype. Thus, the null hypothesis of each experiment that knocking out a single gene *i* has no effect on a single phenotype of interest *j*. As a result, there are 172328 total hypotheses in the dataset. This method of experimentation naturally fits an online structure, because scientists may develop new phenotypes to test against or discover new genes to analyze over time, and consequently want be able to freely extend the set of hypotheses they are testing.

**Results for different alpha-investing algorithms.** In Figure 3, we observe that the only other online FDX control algorithm, LORDFDX, has significantly less power than SupLORD. In fact, LORDFDX is on par or worse than the Bonferroni bound of alpha-spending. Thus, SupLORD is able to obtain more power by delaying control for a constant number of rejections. Similar results are shown in the HMM setting, which we defer, along with additional metrics, to Appendix E. Further, these results are consistent with SupLORD rejecting the most hypotheses on the IMPC data set in Table 2 — the gains in power seem to translate to a real world setting.

Interestingly, these results also suggest a reversal of the previous notion that FDX control was stricter than FDR control. Previously, LORDFDX for controlling FDX at some $\delta^*$ was demonstrated by JM to be less powerful than LORD controlling FDR at a level of $\ell = \delta^*$ in experiments. This is due to LORDFDX being LORD with an additional constraint i.e. if this additional constraint is violated, LORDFDX enters alpha-death. Hence, it is impossible for LORDFDX to make more rejections that LORD when they are controlled at equivalent levels i.e. $\ell = \delta^*$. On the other hand, SupLORD uses the time-uniform bounds from KR to establish control of FDX. SupLORD suggests that controlling FDX is not necessarily more restrictive than controlling FDR. Since high probability bounds may be of greater interest to the user in practice, SupLORD demonstrates that controlling FDX does not necessarily sacrifice power.

Table 2: Comparing total discoveries (rejected hypotheses) by each algorithm on the IMPC data. SupLORD rejects the most hypotheses.

| **Method** | **Rejections** |
|---|---|
| Bonferroni | 879 |
| LORD | 12352 |
| LORDFDX | 8 |
| SupLORD | *18793* |

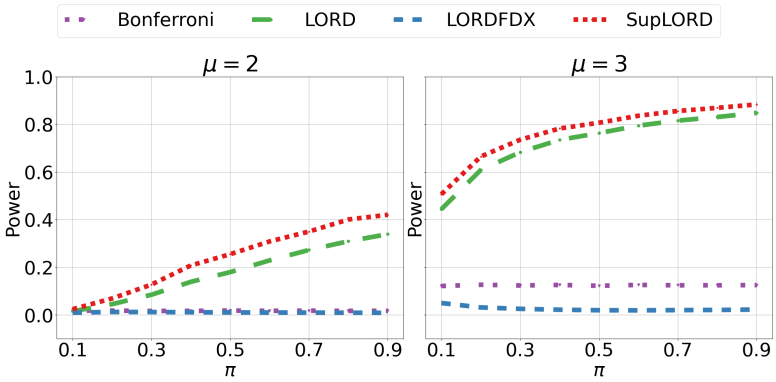



Figure 3: Plot of non-null likelihood, $\pi$, vs. power for signal strengths of $\mu \in \{2,3\}$. FDR is controlled at $\ell = 0.05$, and SupLORD and LORDFDX are controlled at FDX with $\epsilon^* = 0.15$ at a level of $\delta^* = 0.05$. Experiment details are in Section 3.2. SupLORD consistently has higher power across all non-null likelihoods and signal strengths.

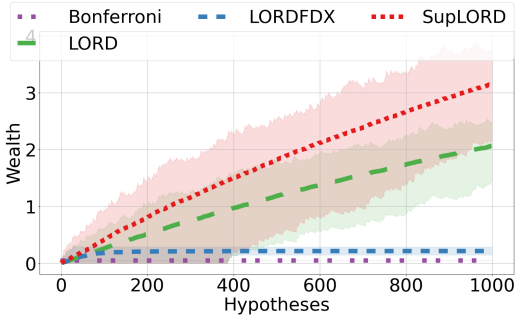
## 4. Dynamic Scheduling With Non-Monotone Spending Rules

**Under-utilized wealth leads to unnecessary conservativeness.** Recall that the wealth value at each step, $W(k)$, quantifies the error budget that can be allocated to the next alpha value. Figure 4(*a*) shows that, empirically, the wealth increases as more hypotheses are tested when using STEADY with SupLORD and LORD. To the best of our knowledge, this problem of STEADY has not been addressed in previous work. The algorithm is not fully utilizing its error budget — the rate at which the algorithm is making rejections and accumulating wealth is much higher than the rate which wealth is spent. While AGGRESSIVE does have better utilization of its wealth, it also does not distribute its spending as evenly as STEADY, since it spends a constant percentage of its current wealth at each step. To remedy this, we derive a new spending schedule that uses wealth much more rapidly when wealth has been accumulated. At the same time, we do not want our algorithm to spend excessively when wealth is scarce. Thus, our new spending schedule is *adaptive* to its wealth — it adjusts its rate of spending to evenly spend as much wealth as possible while avoiding alpha-death.

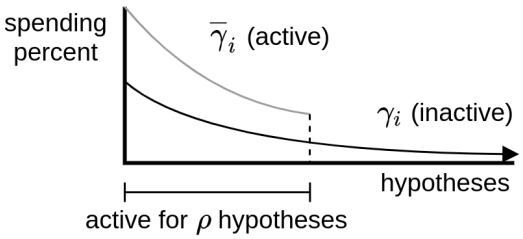
At a technical level, the underutilization of wealth occurs because prior alpha-spending rules are "monotone" (5) functions of past rejections, effectively handicapping them by making them unable to adapt to the current wealth. This monotonicity restriction was primarily imposed in order to mathematically prove FDR control, instead of mFDR control, in a proof technique introduced by Javanmard and Montanari (2018), and extended by all the followup works by Ramdas and coauthors. Our new proof technique avoids the requirement for monotonicity entirely, thus opening the possibility of designing smarter, wealth-adaptive spending rules, while still having FDX and FDR control.

### 4.1. Dynamic Scheduling Formulation

To address this problem, we provide a method for altering $\{\gamma_i\}_{i\in\mathbb{N}}$, called *dynamic scheduling*, based on the wealth of the algorithm. The general paradigm of our new spending schedule is similar to the STEADY spending schedule in (13). In the STEADY spending schedule, we can see that each alpha value is the sum of portions of the wealth boosts at each past rejection. The portion of the wealth boost of the $i$th rejection that the current alpha value spends is dependent on $\gamma_{k-t_i}$, where $k$ is our current time, and $t_i$ is the time the $i$th rejection was made. Our new spending schedule follows the




(*a*) Wealth of different GAI algorithms (setting specified in Section 3.2). The line is the mean over 200 trials and the shaded area is a 95% confidence set. SupLORD and LORD both have increasing wealth.




(*b*) Visualization of $\overline{\gamma}_i$ vs $\gamma_i$ in (16). $\overline{\gamma}_i$ is only active for $\rho$ duration and is much larger than $\gamma_i$ during that duration, which results in increased spending.

Figure 4: Visualization of concepts pertaining to dynamic scheduling.

same paradigm. However, we use a spending sequence that changes based upon the wealth of the algorithm at the time of the rejection, which we denote $\{\overline{\gamma}_i\}_{i\in\mathbb{N}}$. To create this, we must first have a base spending sequence $\{\gamma_i\}_{i\in\mathbb{N}}$. Then, we formulate this new spending sequence as follows:

$$\overline{\gamma}_i(W,W(0)) \equiv \begin{cases} \dfrac{\gamma_i}{\sum_{j=1}^{\rho}\gamma_j}^{\eta\cdot\frac{W}{W(0)}\vee 1} & \text{if } \eta\cdot\frac{W}{W(0)}>1,\, i\le\rho \text{ (active)} \\ 0 & \text{if } \eta\cdot\frac{W}{W(0)}>1,\, i>\rho \text{ (active and finished)} \\ \gamma_i & \text{else (inactive),} \end{cases} \tag{16}$$

where $W$ and $W(0)$ are the wealth of the algorithm at the time of rejection and the beginning, respectively. $\overline{\gamma}_i$ adapts to wealth by switching between the active and inactive cases. If the wealth is high when making a rejection, $\overline{\gamma}_i$ goes into the active case where it spends wealth at a more rapid pace. Otherwise, it falls back to spending according to $\gamma_i$. Figure 4(*b*) illustrates this adaptivity. $\eta\in\mathbb{R}^+$ controls the pace at which $\overline{\gamma}_i$ spends wealth as a function the algorithm's current wealth. $\rho\in\mathbb{N}$ controls the length of time before $\overline{\gamma}_i$ has exhausted the wealth earned from its rejection — the first $\rho$ elements of $\gamma_i$ are normalized to create $\overline{\gamma}_i$. Larger $\eta$ and smaller $\rho$ increases spending in the active case. Note that $\overline{\gamma}_i$ satisfy conditions (12), making it a valid spending sequence.

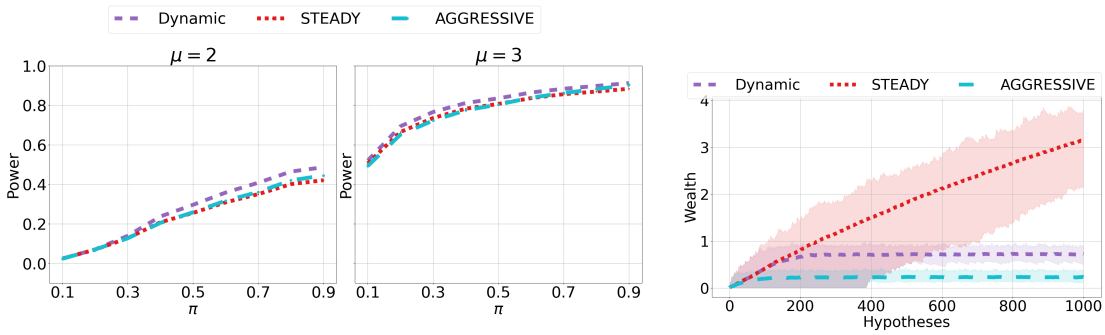
Thus, using $\overline{\gamma}_i$ allows for the spending schedule to quickly use up entire boosts in wealth from rejections if the current wealth of the algorithm is large. We define our dynamic alpha schedule as:

$$\alpha_k = \overline{\gamma}_k(W(0),W(0))\cdot\beta_0 + \sum_{i=1}^{|\mathcal{R}_{k-1}|}\overline{\gamma}_{k-t_i}(W(t_i),W(0))\cdot\beta_{t_i}. \tag{17}$$

We will show through simulation results that this method does solve the problem of wealth underutilization and increases the power of SupLORD as a result.

### 4.2. Numerical Simulations and Real Data Experiments for Dynamic Algorithms

**Spending schedules.** We compare the differences in STEADY, AGGRESSIVE, and dynamic scheduling on SupLORD. We choose $\epsilon^*=0.15$, $\delta^*=0.05$, $r^*=30$ as the parameters for simulations, and




(*a*) Non-null likelihood $\pi$ vs. power (*b*) Wealth vs. time

Figure 5: The left plot compares different spending schedule choices for SupLORD plotted by likelihood of non-null hypotheses, $\pi$, vs. power for signal strengths of $\mu \in \{2,3\}$ in the manner specified in Section 3.2. Dynamic scheduling provides a consistent (albeit small) power improvement over STEADY and AGGRESSIVE SupLORD. The right plot shows that the dynamic scheduling does not hoard wealth like STEADY, but saves more than AGGRESSIVE.

$r^* = 100$ for the IMPC dataset in all SupLORD algorithms. We use the default boost sequence and spending sequence for all three schedules.

**Results for comparing spending schedules.** Figure 5(*a*) shows that using a dynamic schedule does increase power when the signal and likelihood of the non-null hypotheses are larger, and does not do worse than the STEADY or AGGRESSIVE when the signal and likelihood of non-nulls are low. This corresponds to the formulation of the $\overline{\gamma}_i$, as it falls back to $\gamma_i$ if the algorithm's wealth is low. In the largest settings of non-null likelihood and signal, AGGRESSIVE catches up to dynamic scheduling in power. Admittedly, dynamic scheduling is limited in its power gains, since it only allocates existing wealth more effectively. Yet, it still outperforms both baseline schedules across simulation settings — we illustrate that dynamic scheduling generally produces larger alpha values than either baseline schedule in Appendix F. Further, a dynamic schedule also makes more rejections than other schedules on the IMPC dataset in Table 3, which is consistent with its power improvement in simulations. Thus, a dynamic schedule fully utilizes wealth and is a good default for practitioners.

Table 3: Rejections made by each schedule on the IMPC data. Dynamic scheduling has the most rejections.

| **Schedule** | **Rejections** |
|---|---|
| STEADY | 18519 |
| AGGRESSIVE | 15416 |
| dynamic | *18793* |

## 5. Improving FDR Control Through SupFDR Control

Finally, we illustrate the connections between FDR, FDX, and SupFDR. We use additional results from KR to prove that SupLORD controls the SupFDR, and by extension, FDR. These FDR results close the gap between FDR control and mFDR control (4), which was first introduced by FS as a proxy for FDR. FS showed that $\mathrm{mFDR}(\mathcal{R}_\tau)$, where $\tau$ is a stopping time with respect to $\{\mathcal{F}_k\}$, could be controlled under assumption (2). One example of a stopping time is the time of the $k$-th rejection, for some fixed $k \in \mathbb{N}$. Successive works (JM, Ramdas et al. 2017) have managed to show FDR control with more restrictive assumptions of (3) and monotone spending schedules.[3] In addition, they also have only controlled at FDR fixed times, unlike the stopping time control of mFDR proved by FS.

3. Technically, JM do have a theorem that proves FDR control for dependent p-variables, but it is dependent on a strict uniformity assumption and also suffers from alpha-death since the proportion of wealth it can spend at each step is inversely proportional to the logarithm of the number of hypotheses that have been tested.

Thus, SupLORD completes the original goal of FS by showing control, under the same assumptions, on the actual desired metric, FDR, for stopping times. To prove this, we first note SupLORD provides a family of FDX bounds by Fact 1.

**Corollary 3 (Corollary of Theorem 2)** *Assuming conditional superuniformity (2), any GAI algorithm that satisfies conditions (14) also ensures the following family of bounds for any $\delta \in (0,1)$:*

$$\mathrm{FDX}_{t_{r^*}}\left(\frac{\overline{\log}\left(\frac{1}{\delta}\right)}{\overline{\log}\left(\frac{1}{\delta^*}\right)}\epsilon^*\right) \leq \delta.$$

Before we state our control of SupFDR, we will introduce the following constant:

$$c = \int_0^1 \overline{\log}\left(\frac{1}{\delta}\right) d\delta \approx 1.419\,.$$

**Theorem 4** *Assuming conditional superuniformity (2), any GAI algorithm that satisfies conditions (14) ensures the following bound holds:*

$$\mathrm{SupFDR}_{t_{r^*}} \leq \frac{c\epsilon^*}{\overline{\log}\left(\frac{1}{\delta^*}\right)}.$$

A detailed proof is provided in Appendix B of the supplement. Intuitively, we can achieve this expectation by integrating over the family of FDX bounds described in Corollary 3. Thus, this theorem establishes a connection between controlling FDX and SupFDR for SupLORD. Unlike prior FDR bounds which are only valid at fixed times, Theorem 4 provides a bound on FDR for stopping times. Formally, a stopping time $\tau$ is a random variable where the event $\tau = k$ is measurable in $\mathcal{F}_k$ for all $k$. Due to the supremum in Theorem 4, we may derive the following corollary:

**Corollary 5** *Assuming conditional superuniformity (2), $\tau$ is a stopping time that satisfies $\tau \geq t_{r^*}$, any GAI algorithm that satisfies conditions (14) ensures the following bounds hold:*

$$\mathrm{FDR}(\mathcal{R}_\tau) \leq \frac{c\epsilon^*}{\overline{\log}\left(\frac{1}{\delta^*}\right)}.$$

Thus, using SupLORD provides the benefit of maintaining FDR control even when the user may desire to adaptively stop testing hypotheses, and lends flexibility to its usage as a result. Since this result follows directly from Theorem 4, it does not require any extra assumptions besides conditional superuniformity (2). In contrast, prior online FDR methods (Ramdas et al. 2017, 2018, JM) require stronger assumptions of independence (3) and monotone spending schedules (5). Further, they only control FDR at fixed times. Consequently, Corollary 5 improves upon prior work in both the generality of its assumptions and the strength of its guarantee. Additional SupLORD guarantees are discussed in Appendix C (tighter FDR control at fixed times) and Appendix D (mFDR control at stopping times).

## 6. Related Work on Online Multiple Testing

Online multiple testing was introduced by FS (Foster and Stine, 2008), which formulated the problem and setup, and the first algorithm better than a Bonferroni bound through the paradigm of "alpha-investing". Notably, FS introduced the notion of mFDR and proved control at stopping times for mFDR in this work because they found FDR control difficult to prove. Aharoni and Rosset (2014)

extended the guarantees from FS to a broader class of algorithms. The LORD algorithm formulated by JM (Javanmard and Montanari, 2018) is a particularly powerful type of GAI algorithm, and JM showed it could control FDR at all fixed times — this was consequently the first substantial FDR control result in the area. In fact, Chen and Arias-Castro (2020) proved that LORD is asymptotically as powerful as the best offline algorithm for controlling FDR. Ramdas et al. (2017) then relaxed the conditions for FDR control in LORD and formulated a family of algorithms, known as GAI++, that improved power over any GAI algorithm, including LORD. For instances of application, Xu et al. (2015) provide a comprehensive overview of how online hypothesis testing plays a crucial in modern A/B testing. Robertson and Wason (2018); Robertson et al. (2019) apply the problem in a biomedical setting.

Several works have improved the power of the LORD++ algorithm (Tian and Ramdas, 2019; Ramdas et al., 2018). This paper's methods are not specific to LORD — we may generalize dynamic scheduling to any generalized alpha-investing algorithm and the FDX control obtained by SupLORD can also be applied to algorithms with other FDP estimators, like the one in Ramdas et al. (2018).

Other related work on online selective inference includes online methods for controlling family-wise error rate (Tian and Ramdas, 2021), false coverage rate (Weinstein and Ramdas, 2020), and FDR in minibatches i.e. neighboring batches of hypotheses also have FDR control (Zrnic et al., 2020). Controlling any of these metrics guarantees control of the FDR as well. Zrnic et al. (2021) considers FDR control under local dependence between p-values. Gang et al. (2020) take a Bayesian approach to the problem and increase the power of their algorithms by exploiting local structure.

All aforementioned works are primarily concerned with error control in expectation and thus are focused on controlling the FDR. In contrast, this paper introduces an first empirically powerful algorithm, SupLORD, for controlling the FDP at a set level with high probability i.e. FDX. Further, control at stopping times has only been shown for mFDR by FS and Aharoni and Rosset (2014). SupLORD is the first algorithm to demonstrate any control of FDR at stopping times. Thus, SupLORD broadens the online multiple testing literature to account for FDX control and fills the existing gap of controlling FDR at stopping times.

## 7. Summary

We have shown a new algorithm, SupLORD, that controls FDX and is empirically more powerful by a significant amount than previous FDX controlling algorithms. We also show that SupLORD may simultaneously control FDX, FDR, and SupFDR together. In addition, we make the observation that existing algorithms are not tight to their level of error control, and formulate dynamic scheduling, a framework for adapting to the wealth of an algorithm. We empirically show that dynamic scheduling increases power and reduces this wealth gap. Thus, we formulated the most powerful algorithms for FDX control and brought insight to the interplay between multiple error metrics in this paper. Finally, our paper showed the first non-trivial algorithm with FDR control at stopping times, filling a gap in the literature that had been noticed by FS when they first proposed the problem. We plan on implementing our methods and incorporating them into the `onlineFDR` R package (Robertson et al., 2019), which contains current state-of-the-art online multiple testing algorithms. One line for future work is understanding how $\eta$ and $\rho$ can be set optimally in dynamic scheduling, and whether there is choice of parameters that is provably most powerful or optimal in some sense.

**Acknowledgements** We would like to thank our anonymous reviewers for their suggestions and questions. AR acknowledges support from NSF CAREER 1945266.

## References

Ehud Aharoni and Saharon Rosset. Generalized $\alpha$-investing: definitions, optimality results and application to public databases. *Journal of the Royal Statistical Society: Series B: Statistical Methodology*, pages 771–794, 2014.

Shiyun Chen and Ery Arias-Castro. On the power of some sequential multiple testing procedures. *Annals of the Institute of Statistical Mathematics*, April 2020. ISSN 0020-3157, 1572-9052. doi: 10.1007/s10463-020-00752-5.

Dean Foster and Robert A Stine. Alpha-Investing: A Procedure for Sequential Control of Expected False Discoveries. *Journal of the Royal Statistical Society: Series B: Statistical Methodology*, 70(2): 429, 2008.

Bowen Gang, Wenguang Sun, and Weinan Wang. Structure-Adaptive Sequential Testing for Online False Discovery Rate Control. *arXiv:2003.00113 [stat]*, February 2020.

Adel Javanmard and Andrea Montanari. Online rules for control of false discovery rate and false discovery exceedance. *The Annals of Statistics*, 46(2):526–554, 2018.

Natasha A. Karp, Jeremy Mason, Arthur L. Beaudet, Yoav Benjamini, Lynette Bower, Robert E. Braun, Steve D. M. Brown, Elissa J. Chesler, Mary E. Dickinson, Ann M. Flenniken, Helmut Fuchs, Martin Hrabe de Angelis, Xiang Gao, Shiying Guo, Simon Greenaway, Ruth Heller, Yann Herault, Monica J. Justice, Natalja Kurbatova, Christopher J. Lelliott, K. C. Kent Lloyd, Ann-Marie Mallon, Judith E. Mank, Hiroshi Masuya, Colin McKerlie, Terrence F. Meehan, Richard F. Mott, Stephen A. Murray, Helen Parkinson, Ramiro Ramirez-Solis, Luis Santos, John R. Seavitt, Damian Smedley, Tania Sorg, Anneliese O. Speak, Karen P. Steel, Karen L. Svenson, Shigeharu Wakana, David West, Sara Wells, Henrik Westerberg, Shay Yaacoby, and Jacqueline K. White. Prevalence of sexual dimorphism in mammalian phenotypic traits. *Nature Communications*, 8(1):15475, June 2017. ISSN 2041-1723. doi: 10.1038/ncomms15475.

Eugene Katsevich and Aaditya Ramdas. Simultaneous high-probability bounds on the false discovery proportion in structured, regression and online settings. *Ann. Statist.*, 48(6):3465–3487, 12 2020. doi: 10.1214/19-AOS1938. URL https://doi.org/10.1214/19-AOS1938.

Aaditya Ramdas, Fanny Yang, Martin J Wainwright, and Michael I Jordan. Online control of the false discovery rate with decaying memory. In *Advances In Neural Information Processing Systems*, pages 5650–5659, 2017.

Aaditya Ramdas, Tijana Zrnic, Martin Wainwright, and Michael Jordan. SAFFRON: an adaptive algorithm for online control of the false discovery rate. In *International Conference on Machine Learning*, pages 4286–4294, 2018.

David S Robertson and James Wason. Online control of the false discovery rate in biomedical research. *arXiv preprint arXiv:1809.07292*, 2018.

David S. Robertson, Lathan Liou, Aaditya Ramdas, Adel Javanmard, Jinjin Tian, Tijana Zrnic, and Natasha A. Karp. *onlineFDR: Online error control*, 2019. R package 1.7.1.

Jinjin Tian and Aaditya Ramdas. ADDIS: an adaptive discarding algorithm for online FDR control with conservative nulls. In *Advances in Neural Information Processing Systems*, pages 9388–9396, 2019.

Jinjin Tian and Aaditya Ramdas. Online control of the familywise error rate. *Statistical Methods in Medical Research*, 2021. ISSN 0962-2802. doi: 10.1177/0962280220983381.

Larry Wasserman. *All of Statistics: A Concise Course in Statistical Inference*. Springer Science & Business Media, September 2004. ISBN 978-0-387-40272-7.

Asaf Weinstein and Aaditya Ramdas. Online control of the false coverage rate and false sign rate. In *International Conference on Machine Learning*, 2020.

Ya Xu, Nanyu Chen, Addrian Fernandez, Omar Sinno, and Anmol Bhasin. From infrastructure to culture: A/B testing challenges in large scale social networks. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2227–2236, 2015.

Tijana Zrnic, Daniel Jiang, Aaditya Ramdas, and Michael Jordan. The Power of Batching in Multiple Hypothesis Testing. In *International Conference on Artificial Intelligence and Statistics*, pages 3806–3815, 2020.

Tijana Zrnic, Aaditya Ramdas, and Michael I. Jordan. Asynchronous Online Testing of Multiple Hypotheses. *Journal of Machine Learning Research*, 22(33):1–39, 2021. ISSN 1533-7928.

## Appendix A. SupLORD General Formulation

We first define a more general form of SupLORD that is valid for any $\gamma^*, r^*$. We add an additional user parameter of $a > 0$. We now redefine our estimators of FDP to include $a$.

$$\widehat{\mathrm{FDP}}_a(\mathcal{R}_k) \equiv \frac{\widehat{V}_k + a}{|\mathcal{R}_k|},$$
$$\overline{\log}_{[a]}\left(\frac{1}{\delta}\right) \equiv \frac{\log(\frac{1}{\delta})}{a\log\left(1+\frac{\log(\frac{1}{\delta})}{a}\right)},$$
$$\overline{\mathrm{FDP}}_a(\mathcal{R}_k) \equiv \overline{\log}_{[a]}\left(\frac{1}{\delta}\right) \cdot \widehat{\mathrm{FDP}}_a(\mathcal{R}_k).$$

We drop $a$ from the notation when $a = 1$. We first cite the more general form of Fact 1.

**Fact 2 (Detailed form of Theorem 4 from KR)** *Let $\{\alpha_k\}_{k\in\mathbb{N}}$ be any sequence of alpha values predictable with respect to filtration $\mathcal{F}_k$ in definition (1). Assuming conditional superuniformity (2), the following uniform bound holds:*

$$\mathbf{Pr}\big(\mathrm{FDP}(\mathcal{R}_k) \le \overline{\mathrm{FDP}}_a(\mathcal{R}_k) \text{ for all } k \ge 1\big) \ge 1-\delta.$$

Consequently, our conditions on the boost sequence $\{\beta_i\}_{i\in\mathbb{N}\cup\{0\}}$ is modified to account for $a$:

$$\sum_{i=0}^{t_{r^*-1}} \beta_i R_i \le \frac{\epsilon^* r^*}{\overline{\log}_{[a]}\left(\frac{1}{\delta^*}\right)} - a \tag{18a}$$
$$\beta_i \le \frac{\epsilon^*}{\overline{\log}_{[a]}\left(\frac{1}{\delta^*}\right)} \text{ for all } i > t_{r^*-1}. \tag{18b}$$

The default $\{\beta_i\}_{i\in\mathbb{N}\cup\{0\}}$ can be specified as follows:

$$\beta_i = \begin{cases} \frac{1}{r^*}\left(\frac{\epsilon^* r^*}{\overline{\log}_{[a]}\left(\frac{1}{\delta^*}\right)} - a\right) & i < r^*, \\ \frac{\epsilon^*}{\overline{\log}_{[a]}\left(\frac{1}{\delta^*}\right)} & i \ge r^* \end{cases}. \tag{19}$$

Choosing $a = 1$ recovers conditions (14) and the default sequence in (15).

Naturally, the we then prove the following bound holds for the general form:

**Theorem 6** *Given that conditional superuniformity (2) holds, let $0 < \epsilon^*, \delta^* < 1$, $r^* \ge 1$ be a positive integer, and $t_{r^*}$ be a random variable that is the time when the $r^*$th rejection is made. A GAI algorithm with a boost sequence $\{\beta_i\}_{i\in\mathbb{N}\cup\{0\}}$ that satisfies conditions (18) guarantees:*

$$\mathrm{FDX}_{t_{r^*}}^{\epsilon^*} \le \delta^*.$$

Similar to Section 3, Theorem 6 follows from Fact 2 and a variant of Lemma 1 for the general form SupLORD — Lemma 7. We prove Lemma 7 in the sequel.

### A.1. Proof of Lemma 1

First, we introduce this general form version of Lemma 1:

**Lemma 7** *An GAI algorithm with a boost sequence* $\{\beta_i\}_{i\in\mathbb{N}\cup\{0\}}$ *that satisfies conditions* (18) *will ensure the following:*

$$\overline{\mathrm{FDP}}(\mathcal{R}_k)\le\epsilon^*$$

*for all* $k\ge t_{r^*}$ *such that the* $k$*th hypothesis is rejected.*

**Proof** Let $k=t_r$ where $r\ge r^*$. $|\mathcal{R}_{t_r-1}|=|\mathcal{R}_{t_r}|-1$ since a new rejection is made at $t_r$:

$$\begin{aligned}
\overline{\mathrm{FDP}}_a(\mathcal{R}_{t_r}) &= \overline{\log}_{[a]}\left(\frac{1}{\delta^*}\right)\cdot\left(\frac{\sum_{i=1}^{t_r}\alpha_i+a}{|\mathcal{R}_{t_r}|}\right) && \text{by definition}\\
&= \overline{\log}_{[a]}\left(\frac{1}{\delta^*}\right)\cdot\left(\frac{\beta_0+\sum_{i=1}^{t_r-1}\beta_i-W(k-1)+\alpha_{t_r}+a}{|\mathcal{R}_{t_r}|}\right) && \text{by (10)}\\
&\le \overline{\log}_{[a]}\left(\frac{1}{\delta^*}\right)\cdot\left(\frac{\beta_0+\sum_{i=1}^{t_r-1}\beta_i+a}{|\mathcal{R}_{t_r}|}\right) && \text{by (11)}\\
&= \overline{\log}_{[a]}\left(\frac{1}{\delta^*}\right)\cdot\left(\frac{\frac{\epsilon^* r^*}{\overline{\log}_{[a]}\left(\frac{1}{\delta^*}\right)}+\frac{\epsilon^*(|\mathcal{R}_{t_r-1}|-(r^*-1))}{\overline{\log}_{[a]}\left(\frac{1}{\delta^*}\right)}}{|\mathcal{R}_{t_r}|}\right) && \text{by conditions (18)}\\
&\le \overline{\log}_{[a]}\left(\frac{1}{\delta^*}\right)\cdot\left(\frac{\frac{\epsilon^* r^*}{\overline{\log}_{[a]}\left(\frac{1}{\delta^*}\right)}+\frac{\epsilon^*(|\mathcal{R}_{t_r}|-1-(r^*-1))}{\overline{\log}_{[a]}\left(\frac{1}{\delta^*}\right)}}{|\mathcal{R}_{t_r}|}\right) && \text{since a rejection is made at time } t_r\\
&= \epsilon^*.
\end{aligned}$$

Thus, we have proven Lemma 7. ■

Lemma 1 follows directly from setting $a=1$.

### A.2. Tradeoffs in the choice of $a$

Let $w_0$ be the upper bound in condition (18a) and $b$ be the upper bound in condition (18b):

$$\begin{aligned}
w_0 &\equiv \frac{\epsilon^* r^*}{\overline{\log}_{[a]}\left(\frac{1}{\delta^*}\right)}-a,\\
b &\equiv \frac{\epsilon^*}{\overline{\log}_{[a]}\left(\frac{1}{\delta^*}\right)} \text{ for all } i>t_{r^*-1}.
\end{aligned}$$

The offset, $a$, in $\overline{\text{FDP}}_a$ plays a crucial role in determining the tradeoff between $w_0$ and the bound on earning per rejection from $r^*$ onwards, $b$. Given the user chosen parameters $\epsilon^*$, $\delta^*$, and $r^*$, we provide a principled way of choosing $a$ such that both quantities can be maximized.

First, note that the upper bound in condition (18b), $b$, is an increasing function of $a$ that asymptotically approaches $\epsilon^*$. Thus, we cannot maximize it. However, we may maximize $w_0$, the upper bound in condition (18a), which is equivalent to solving the following optimization problem:

$$\max_{a>0} \frac{\epsilon^* r^*}{\overline{\log}_{[a]}\left(\frac{1}{\delta^*}\right)} - a\,.$$

Since the objective is a smooth concave function with respect to $a$, by taking the derivative of the objective and solving for 0, we can determine the maximum $w_0$ is achieved when the following condition holds:

$$\log\left(1+\frac{\log\left(\frac{1}{\delta^*}\right)}{a}\right) - \frac{\log\left(\frac{1}{\delta^*}\right)}{a+\log\left(\frac{1}{\delta^*}\right)} = \frac{\log\left(\frac{1}{\delta^*}\right)}{\epsilon^* r^*}. \tag{20}$$

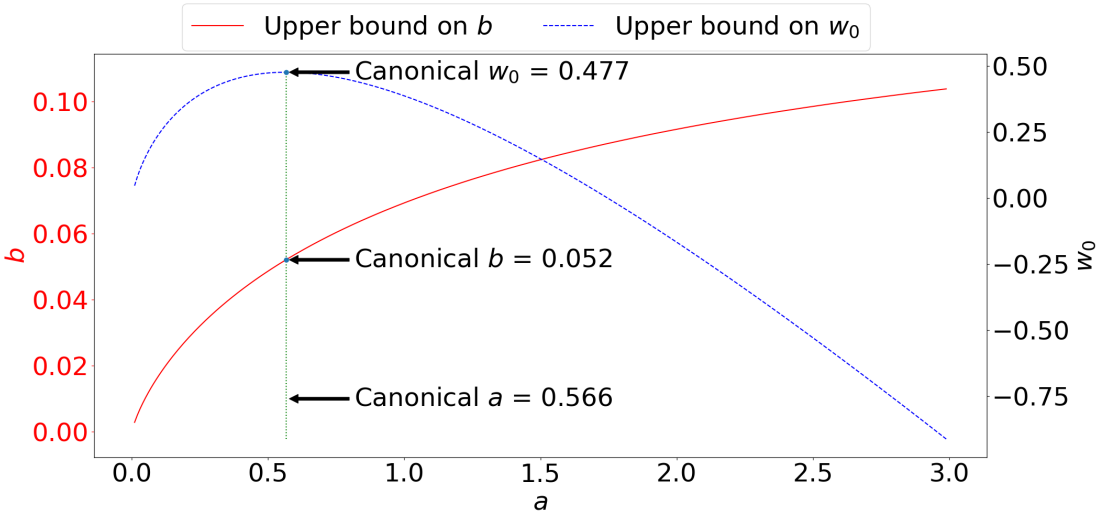



Figure 6: Tradeoff between $w_0$ and $b$ as a function of $a$ given $\delta^*=0.05, \epsilon^*=0.15, r^*=20$. Note that $b$ is increasing with respect to $a$, while $w_0$ has a maximum. We also denote the canonical triple as a triple $(a,b,w_0)$, that can be selected to simultaneously maximize $b$ and $w_0$.

We illustrate in Figure 6 the relationship between $a, b$, and $w_0$ for an example setting of $\epsilon^* = 0.15, \delta^*=0.05$, and $r^*=20$. We define the canonical triple for $(a,b,w_0)$ as the $b$ and $w_0$ corresponding to the $a$ that is the positive solution to problem (20). Note that for all $a$ smaller than the canonical $a$, we may strictly improve any possible $b$ and $w_0$ under that $a$ by selecting the canonical triple. On the other hand, the $b$ and $w_0$ for any $a$ larger than the canonical $a$ form an optimal Pareto-frontier for trading off $b$ and $w_0$. Thus, the canonical $a$ can be treated as the default choice for $a$ and consequently, the user may avoid tuning additional hyperparameters. In addition, we may also define a default $\{\beta_i\}_{i\in\mathbb{N}\cup\{0\}}$ w.r.t. the canonical triple:

$$\beta_i = \begin{cases} \frac{1}{r^*}\left(\frac{\epsilon^* r^*}{\overline{\log}_{[a]}\left(\frac{1}{\delta^*}\right)} - a\right) & i \le t_{r^*-1} \\ \frac{\epsilon^*}{\overline{\log}_{[a]}\left(\frac{1}{\delta^*}\right)} & i > t_{r^*-1}. \end{cases} \tag{21}$$

This default $\{\beta_i\}_{i\in\mathbb{N}\cup\{0\}}$ is tight with conditions (18) for the canonical $a$, and can be considered as the default $\{\beta_i\}_{i\in\mathbb{N}\cup\{0\}}$ sequence for the general SupLORD form.

## Appendix B. Proof of Theorem 4

First, we define the following term:

$$c_a = \int_0^1 \overline{\log}_{[a]}\left(\frac{1}{\delta}\right) d\delta.$$

Note that the $c$ in Section 5 is defined with $a=1$, in which case $c=c_1\leq 1.42$.

**Theorem 8** *Assuming conditional superuniformity (2), any GAI algorithm that satisfies conditions (18):*

$$\mathrm{SupFDR}_{t_{r^*}} \leq \frac{c_a\epsilon^*}{\overline{\log}\left(\frac{1}{\delta^*}\right)}.$$

Theorem 4 can be seen as an instance of this theorem where $a=1$.

To prove this theorem, we must prove a preliminary fact.

**Fact 3 (Corollary 1 from an earlier revision of KR)** *Let $\{\alpha_k\}_{k\in\mathbb{N}}$ be any sequence of alpha values predictable with respect to filtration $\mathcal{F}_k$ in definition (1). Assume conditional superuniformity (2) is true, and let $a>0$ be fixed. Then, we obtain the following expected supremum ratio bound:*

$$\mathbb{E}\left[\sup_{k\in\mathbb{N}} \frac{\mathrm{FDP}(\mathcal{R}_k)}{\widehat{\mathrm{FDP}}_a(\mathcal{R}_k)}\right] \leq c_a.$$

We include a proof of this fact for completeness. First, we introduce another lemma from KR:

**Fact 4 (Lemma 1 from KR)** *Let the conditions of Fact 3 be satisfied. Then, the following is true:*

$$\mathbf{Pr}\left(\sup_{k\in\mathbb{N}} \frac{\mathrm{FDP}(\mathcal{R}_k)}{\widehat{\mathrm{FDP}}_a(\mathcal{R}_k)} > x\right) \leq \exp(-a\theta_x x),$$

*where $\theta_x$ is the unique solution to $\exp(\theta)=1+\theta x$.*

**Proof** [Proof of Fact 3] First, we characterize $x$ when $\exp(-a\theta_x x)=\delta$ in Fact 4:

$$\begin{aligned}
\theta_x x &= -\frac{\log(\delta)}{a}, \\
\theta_x &= \log\left(-\frac{\log(\delta)}{a}+1\right). && \text{by } \exp(\theta_x)=1+\theta_x x
\end{aligned}$$

We now substitute this equality for $\theta_x$ into the following:

$$\begin{aligned}
\exp(\theta_x) &= 1+\theta_x x, \\
x &= \frac{\exp(\theta_x)-1}{\theta_x} \\
&= \frac{\frac{-\log(\delta)}{a}}{\log\left(-\frac{\log(\delta)}{a}+1\right)} \\
&= \frac{\log\left(\frac{1}{\delta}\right)}{a\log\left(1+\frac{\log\left(\frac{1}{\delta}\right)}{a}\right)} \\
&= \overline{\log}_{[a]}\left(\frac{1}{\delta}\right).
\end{aligned}$$

Now, we bound the expectation:

$$\begin{aligned}
\mathbb{E}\left[\sup_{k\in\mathbb{N}} \frac{\mathrm{FDP}(\mathcal{R}_k)}{\widehat{\mathrm{FDP}}_a(\mathcal{R}_k)}\right] &= \int_0^\infty \mathbf{Pr}\left(\sup_{k\in naturals} \frac{\mathrm{FDP}(\mathcal{R}_k)}{\widehat{\mathrm{FDP}}_a(\mathcal{R}_k)} > x\right) dx \\
&\qquad \text{by quantile formula for expectation} \\
&\leq \int_0^\infty \exp(-a\theta_x x)\, dx \qquad \text{by Fact 4} \\
&= \int_1^0 \delta\, d\overline{\log}_{[a]}\left(\frac{1}{\delta}\right) \qquad \text{substitute in } \delta \text{ and } \exp(-a\theta_x x) \\
&= 0\cdot\overline{\log}_{[a]}(0) - \overline{\log}_{[a]}(1) + \int_0^1 \overline{\log}_{[a]}\left(\frac{1}{\delta}\right) d\delta.
\end{aligned}$$

To simplify the last expression, we observe the following two facts:

$$\begin{aligned}
\lim_{\delta\to 0^+} \delta\cdot\overline{\log}_{[a]}\left(\frac{1}{\delta}\right) &= 0, \\
\lim_{\delta\to 1^-} \delta\cdot\overline{\log}_{[a]}\left(\frac{1}{\delta}\right) &= 1.
\end{aligned}$$

Thus, we can drop $0\cdot\overline{\log}_{[a]}(0)$, since it is 0, and $-1\cdot\overline{\log}_{[a]}(1)$ since it negative, and we achieve our desired upper bound. ■

We proceed to prove Theorem 8.

$$\begin{aligned}
\text{SupFDR}_{t_{r^*}} &= \mathbb{E}\left[\sup_{k \geq t_{r^*}} \text{FDP}(\mathcal{R}_k)\right] \\
&= \mathbb{E}\left[\sup_{r \geq r^*} \text{FDP}(\mathcal{R}_{t_r})\right] \\
&\leq \mathbb{E}\left[\sup_{r \geq r^*} \frac{\text{FDP}(\mathcal{R}_{t_r})}{\widehat{\text{FDP}}_a(\mathcal{R}_{t_r})}\right] \cdot \left(\sup_{r \geq r^*} \widehat{\text{FDP}}_a(\mathcal{R}_{t_r})\right) && \text{relaxation of sup} \\
&\leq \frac{c_a}{\overline{\log}_{[a]}\left(\frac{1}{\delta^*}\right)} \cdot \left(\sup_{r \geq r^*} \overline{\text{FDP}}_a(\mathcal{R}_k)\right) && \text{by Fact 3 and definition of } \widehat{\text{FDP}}_a(\mathcal{R}_k) \\
&\leq \frac{c_a \epsilon^*}{\overline{\log}_{[a]}\left(\frac{1}{\delta^*}\right)}. && \text{by Lemma 7}
\end{aligned}$$

Thus, we have shown that the general SupLORD algorithm controls SupFDR as claimed.

## Appendix C. Tighter Fixed Time FDR Control

A tighter, fixed time FDR bound exists for SupLORD with additional constraints:

**Theorem 9** *Assuming independent superuniformity* (3)*, any GAI algorithm with a monotone spending schedule* (5) *that satisfies condition* (18) *and*

$$\beta_0 + \beta_1 \leq \frac{\epsilon^*}{\overline{\log}_{[a]}\left(\frac{1}{\delta^*}\right)}, \tag{22}$$

*can ensure the following fixed time* FDR *bound:*

$$\text{FDR}_1 \leq \frac{\epsilon^*}{\overline{\log}_{[a]}\left(\frac{1}{\delta^*}\right)}.$$

This result uses the same proof techniques as Ramdas et al. (2017, 2018); Tian and Ramdas (2019) for FDR control, hence the much stronger assumptions and control only at fixed times. Thus, we can improve our FDR control by a factor $c_a$ of the cost of usability — independence is not necessarily a practical assumption, and control only at fixed times denies the flexibility of early stopping to the user.

**Proof** To prove Theorem 9, we utilize the following theorem from Ramdas et al. (2017):

**Fact 5 (Theorem 1 from Ramdas et al. 2017)** *Assuming independent superuniformity* (3)*, any GAI algorithm with a monotone spending schedule eq.* (5) *with boost sequence* $\{\beta\}_{i \in \mathbb{N} \cup \{0\}}$ *controls* FDR *at any fixed time:*

$$\text{FDR}_1 \leq \max(\{\beta_0 + \beta_1\} \cup \{\beta_i\}_{i \in \mathbb{N} \cup \{0\}}).$$

Thus, Theorem 9 follows directly from condition (24) and Fact 5. ■

An example specification for $\{\beta_i\}_{i\in\mathbb{N}\cup\{0\}}$ that satisfies the conditions of Theorem 9 is as follows:

$$\beta_i = \begin{cases} \frac{1}{2r^*}\left(\frac{\epsilon^* r^*}{\overline{\log}_{[a]}\left(\frac{1}{\delta^*}\right)} - a\right) & i = 0 \\ \frac{1}{2r^*}\left(\frac{\epsilon^* r^*}{\overline{\log}_{[a]}\left(\frac{1}{\delta^*}\right)} - a\cdot\mathbf{I}\{r^* > 1\}\right) & i \le t_1 \\ \frac{1}{r^*}\left(\frac{\epsilon^* r^*}{\overline{\log}_{[1]}\left(\frac{1}{\delta^*}\right)} - a\right) & t_1 < i \le t_{r^*-1} \\ \frac{\epsilon^*}{\overline{\log}_{[1]}\left(\frac{1}{\delta^*}\right)} & i > t_{r^*-1} \end{cases}. \tag{23}$$

We simply reduce $\beta_0+\beta_1$ such that they sum to at most the upper bound in condition (18b).

## Appendix D. mFDR Control for SupLORD

**Fact 6 (Theorem 2 from Zrnic et al. 2021)** *Let $\tau$ be a bounded stopping time. Assuming conditional superuniformity (2), any GAI algorithm with boost sequence $\{\beta_i\}_{i\in\mathbb{N}\cup\{0\}}$ ensures the following:*

$$\mathrm{mFDR}(\mathcal{R}_\tau) \le \max(\{\beta_0+\beta_1\}\cup\{\beta_i\}_{i\in\mathbb{N}\cup\{0\}}).$$

Essentially, Zrnic et al. (2021) prove that the same algorithm can control FDR at fixed times under p-variable independence and monotone scheduling, and mFDR at stopping times under conditional superuniformity. Thus, we derive mFDR control for SupLORD:

**Theorem 10** *Let $\tau$ be a bounded stopping time. Assuming conditional superuniformity (2), any GAI that satisfies condition (18) and*

$$\beta_0+\beta_1 \le \frac{\epsilon^*}{\overline{\log}_{[a]}\left(\frac{1}{\delta^*}\right)}, \tag{24}$$

*can ensure the following FDR bound:*

$$\mathrm{mFDR}(\mathcal{R}_\tau) \le \frac{\epsilon^*}{\overline{\log}_{[a]}\left(\frac{1}{\delta^*}\right)}.$$

Theorem 10 follows directly from conditions (18) and from Fact 6. We can use the boost sequence in (23) as an example boost sequence that would give SupLORD mFDR control, since it also suffices for the assumptions in Theorem 10.

## Appendix E. Additional Numerical Simulations

### E.1. FDR and SupFDR in FDX Control Simulations

Figure 7 shows that despite the nonexistent power LORDFDX possessed relative to SupLORD, its error in FDR is surprisingly close to SupLORD. On other hand, LORDFDX has very low SupFDR in Figure 8.

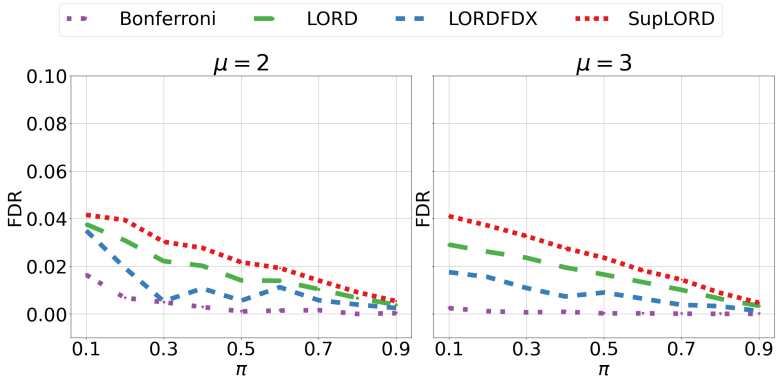



Figure 7: Plot of non-null likelihood, $\pi$, vs. FDR for signal strengths of $\mu \in \{2,3\}$ in the constant data setting. FDR is controlled at $\ell = 0.05$, and SupLORD and LORDFDX are controlled at FDX with $\epsilon^* = 0.15$ at a level of $\delta^* = 0.05$. Experiment details in Section 3.2. SupLORD has FDR that is somewhat appropriate to its results in Theorem 9.

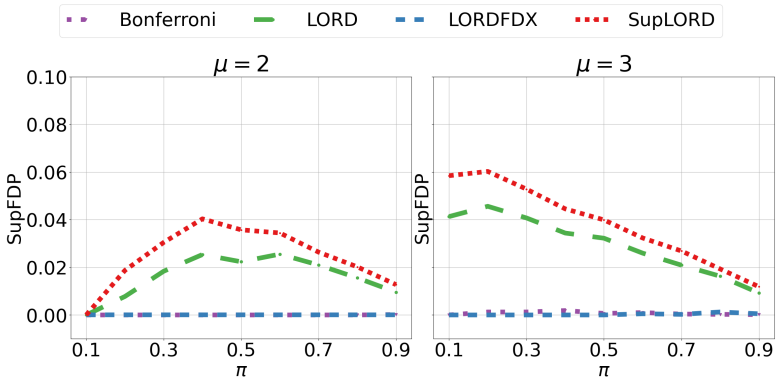



Figure 8: Plot of non-null likelihood, $\pi$, vs. $\text{SupFDR}_{30}$ for signal strengths of $\mu \in \{2,3\}$ in the constant data setting. FDR is controlled at $\ell = 0.05$, and SupLORD and LORDFDX are controlled at FDX with $\epsilon^* = 0.15$ at a level of $\delta^* = 0.05$. Experiment details in Section 3.2. SupLORD has $\text{SupFDR}_{30}$ below what is expected by Theorem 4.

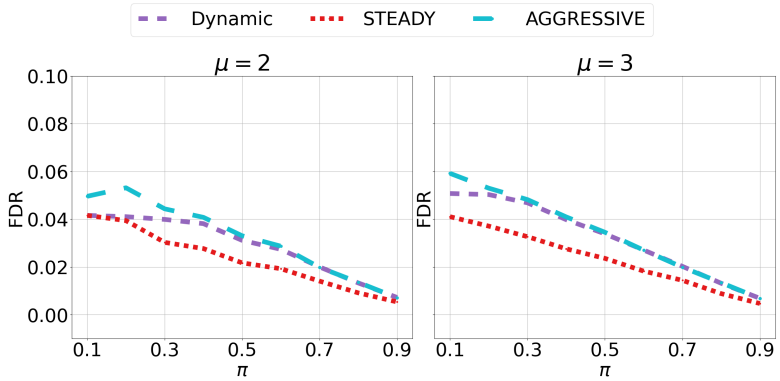



Figure 9: Plot of non-null likelihood, $\pi$, vs. FDR for signal strengths of $\mu \in \{2,3\}$ in the constant data setting. SupLORD is controlled at FDX with $\epsilon^* = 0.15$ at a level of $\delta^* = 0.05$. Data details in Section 3.2, experiment details in Section 4.2. All schedules have similar FDR.

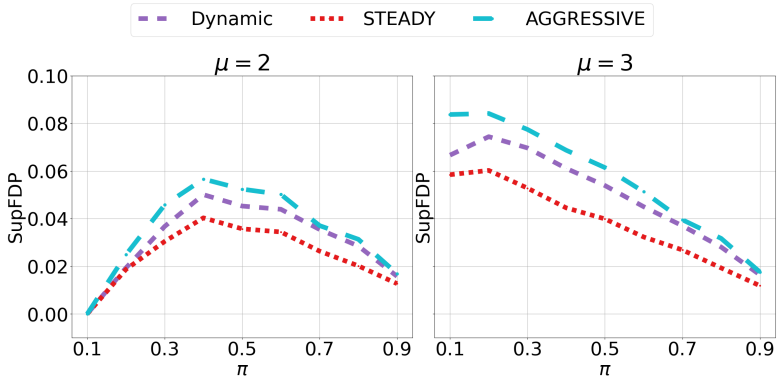



Figure 10: Plot of non-null likelihood, $\pi$, vs. $\text{SupFDR}_{30}$ for signal strengths of $\mu \in \{2,3\}$ in the constant data setting. FDR is controlled at $\ell = 0.05$, and SupLORD and LORDFDX are controlled at FDX with $\epsilon^* = 0.15$ at a level of $\delta^* = 0.05$. Data details in Section 3.2, experiment details in Section 4.2. All schedules have similar SupFDR.

### E.2. Hidden Markov Model (HMM)

In addition to the standard Gaussian setting, we also perform simulations using a HMM process to model a setting where non-null where each hypothesis not independent of neighboring hypotheses. We set $\mu_i = \mu$ to be constant for all $i$, but we define $\pi_i$ as follows:

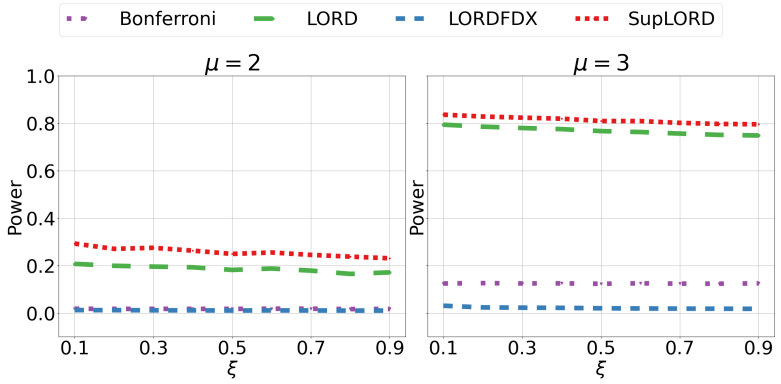
$$\pi_1 = 0.5, \qquad \pi_i = \begin{cases} \xi & \text{if } H_{i-1} \in \mathcal{H}_0, \\ 1-\xi & \text{if } H_{i-1} \notin \mathcal{H}_0. \end{cases}$$

If we let $\beta_i$ be the state at the $i$th time step, we can view the data generating process as a two state HMM model with a $\xi$ probability of transitioning to the other state at the next time step. In expectation, there is an equal number of hypotheses that are null and non-null, regardless of the choice of $\xi$. However, a smaller $\xi$ will result in large clusters of null or non-null hypotheses in consecutive time steps.

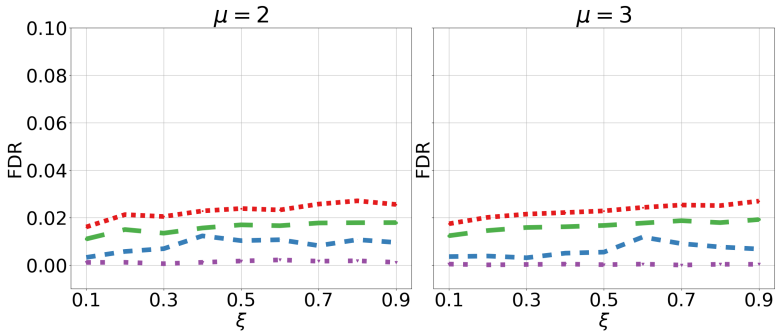
### E.3. Results on HMM

We show extra results of the SupLORD comparison with LORD and LORDFDX on the HMM model in Figure 11.

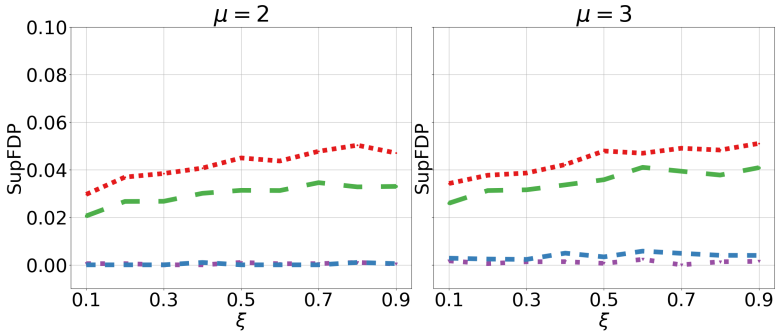
In Figure 12, we observe that in the HMM setting, dynamic scheduling remains the best choice, as it is superior or equal to the other schedules in power on all signal strengths and transition probabilities. In addition, if we compare the performance of dynamic scheduling and AGGRESSIVE we note that the gap between the power of these two schedules is virtually nonexistent when the probability of changing states, $\xi$, is low. The exact reverse is true when comparing dynamic scheduling and STEADY — they have similar power when $\xi$ is high. This reflects the ability of dynamic scheduling to adapt to the distribution of non-null hypotheses by dynamically adjusting its behavior in accordance with the wealth of the algorithm. Consequently, it incorporates the strengths of both STEADY and AGGRESSIVE, and is more powerful regardless of $\xi$ as a result.




(*a*)

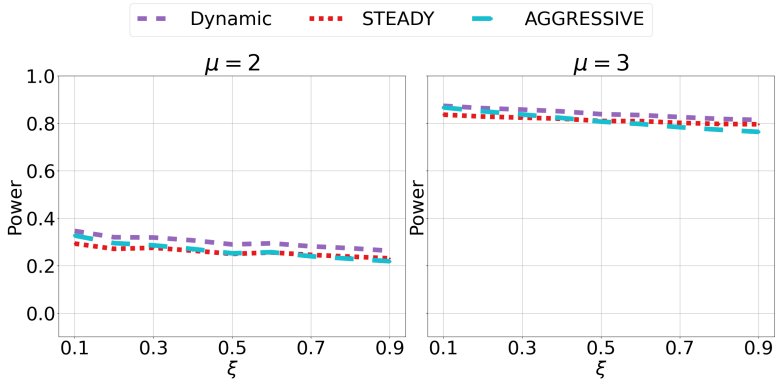



(*b*)

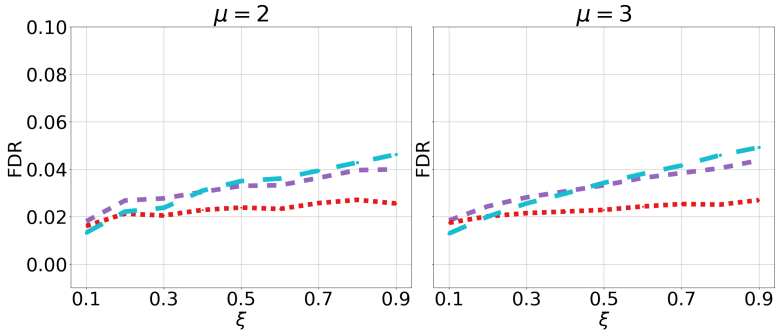



(*c*)

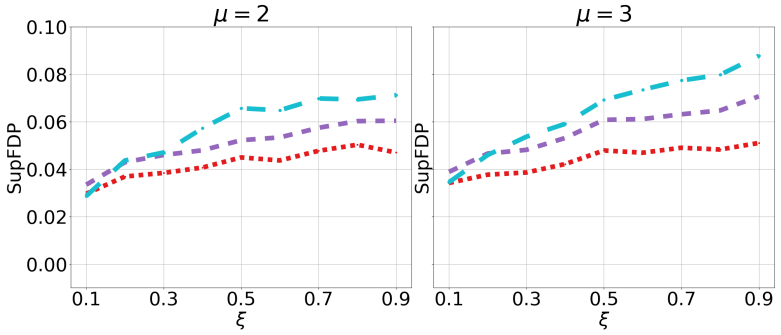
Figure 11: Plot of state transition probability, $\xi$, vs. FDR, $\mathrm{SupFDR}_{30}$, and power where $\ell = 0.05$ for signal strengths of $\mu \in \{2,3\}$ in the HMM data setting. SupLORD and LORDFDX are calibrated with $\delta^* = 0.05$ for $\epsilon^* = 0.15$. We observe that SupLORD has higher power across all $\xi$ and $\mu$, showing that it is not penalized by clustering of null or non-null hypotheses, relative to prior methods. (Recall that the definition of power only considers correctly rejected hypotheses. The gain in power is due to a less conservative algorithm that makes better use of its FDR/FDX budget.)




(*a*)




(*b*)

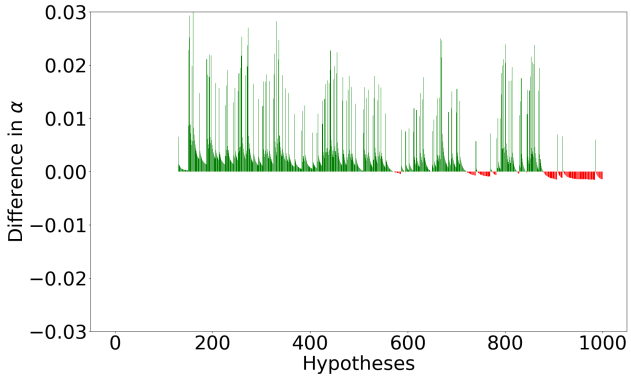



(*c*)

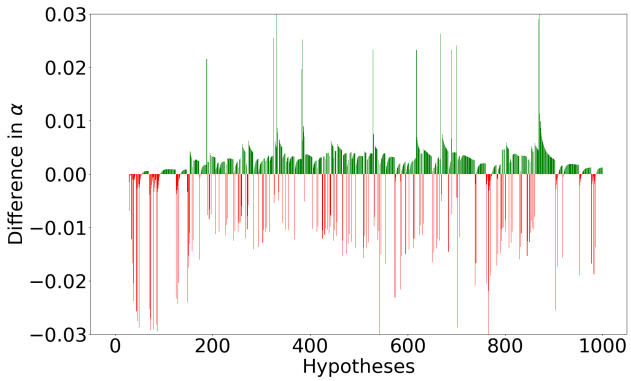
Figure 12: Plot of the state change probability, $\xi$, vs. FDR, $\text{SupFDR}_{30}$, and power in the HMM data setting comparing between SupLORD with dynamic scheduling and without. The power plots suggests that dynamic scheduling is uniformly better than either STEADY or AGGRESSIVE.

## Appendix F. Illustrating the Dynamic Scheduling Improvement

To illustrate the difference between dynamic scheduling and baseline static schedules, we plot the different alpha values produced by each scheduling algorithm with SupLORD for a single trial. In Figure 13, we plot the difference between alpha values of dynamic scheduling and each of the baseline schedules for a single trial in the constant data setting, where non-null frequency is $\pi = 0.3$ and non-null mean is $\mu = 3$. We see that for the majority of hypotheses, dynamic scheduling outputs $\alpha_i$ that are larger by about 0.005. Note that a dynamic schedule only makes more rejections if the p-value lies in the gap between the alpha value output by the baseline schedule, and the alpha value output by the dynamic schedule. Further, a p-value from a non-null hypothesis generally lies in this alpha value gap with slightly higher probability than a uniform distribution for the null hypothesis e.g. it may be within the gap with 0.01–0.02 probability. Thus, dynamic scheduling gets consistent power gains over each of the baselines by having generally larger alpha values than baseline schedules.




(*a*) Dynamic $\alpha$ – STEADY $\alpha$




(*b*) Dynamic $\alpha$ – AGGRESSIVE $\alpha$

Figure 13: Plot of differences in alpha values between dynamic scheduling and each baseline static schedule for an identical trial of p-values generated by the same seed in the constant data setting with non-null frequency $\pi = 0.3$, and non-null mean $\mu = 3$. Green bars mark steps where dynamic scheduling has higher alpha value, and red marks steps where the baseline static schedule has a higher alpha value. Dynamic scheduling has higher alpha values by about 0.005 for the majority of the trial when compared to both baseline schedules.

We plot the actual alpha values for each schedule of the single run in Figure 14 and the mean alpha values of each schedule across 200 runs in Figure 15. In Figure 15, we can observe that dynamic scheduling has consistently larger alpha values than both schedules, albeit by a small amount.

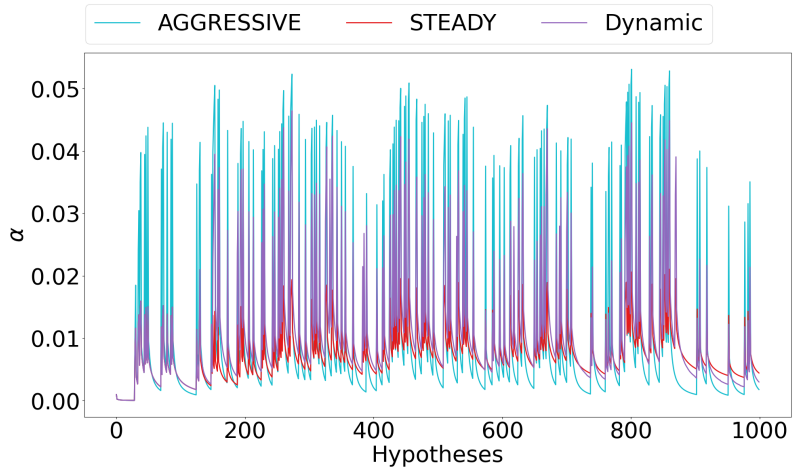



Figure 14: (Raw values) Plot of actual alpha values between dynamic scheduling and each baseline static schedule for an identical trial of p-values generated by the same seed in the constant data setting with non-null frequency $\pi = 0.3$, and non-null mean $\mu = 3$.

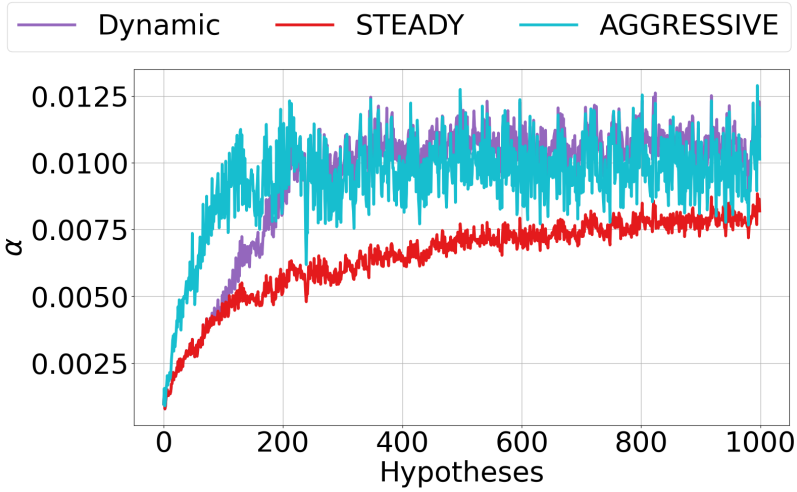



Figure 15: (Mean values) Plot of mean alpha values between dynamic scheduling and each baseline static schedule in the constant data setting with non-null frequency $\pi = 0.3$, and non-null mean $\mu = 3$ over 200 trials. We can observe that dynamic scheduling is generally larger than both baseline schedules by a small amount in a consistent fashion.