Fully Specified Estimation Plan for Optimal Static Parametric Estimation of Arbitrary Distributions (OSPEAD)

Public Version

Rucknium

November 18, 2022

1 Introduction

Research on Monero and similar systems has established the vital importance of constructing a decoy selection algorithm that closely resembles the real spend age distribution. A sample of researchers' conclusions and recommendations follows:

> As we have seen the current [2017] sampling strategy for mix-ins fails drastically in preventing temporal analysis. There are two possible strategies towards mitigating the ensuing risks: (a) mimic users' spending behavior or, (b) force mix-ins to be picked according to some "unknown" distribution.
>
> - [Kumar et al., 2017]

> We have provided evidence that the strategy by which Monero mixins are sampled [circa 2017] results in a different time distribution than real spends, significantly undermining the untraceability mechanism. To correct this problem, the mixins should ideally be sampled in such a way that the resulting time distributions match.
>
> - [Möser et al., 2018]

> The mixin sampling distribution has since been replaced with a gamma distribution (from [Möser et al., 2018]) fitted to the empirical spend-time distribution. Our results show that these sampling distribution changes have made a significant impact in reducing the accuracy of the guess-newest heuristic.
>
> - [Ye et al., 2020]

> [A mimicking decoy selection algorithm's] anonymity depends on how well $\hat{\mathcal{S}}$ [the decoy selection algorithm] estimates $\mathcal{S}$ [the real spend age distribution]....It is therefore reasonable to expect that if the mimicking sampler has access to the true source distribution $\mathcal{S}$, its anonymity should be close to optimal. In the following, we give an [sic] evidence that this is the case.
>
> - [Ronge et al., 2021]

The problem of creating a decoy selection algorithm that minimizes the usefulness of timing information to an adversary was recognized by the Monero Research Lab very early [Mackenzie et al., 2015]:

> One solution to this problem is to determine a non-uniform method of choosing transaction outputs for ring signatures; choose transaction outputs based on their age such that the probability that these outputs are chosen for a ring signature is inversely related to the probability that they have been spent already. This would suggest that we, the developers of Monero, must estimate the probability distribution governing the age of transaction outputs.

A reliable estimator of the real spend age probability distribution has been elusive. [Möser et al., 2018] estimated the distribution based on partial knowledge of real spends obtained from other de-anonymization techniques. The introduction of RingCT and other improvements to Monero have rendered the de-anonymization techniques of [Möser et al., 2018] ineffective ([Ye et al., 2020], [Vijayakumaran, 2021]). Therefore, an updated and improved estimate would require an estimator to use only the fully anonymized ring data on the Monero blockchain.

In this document I lay out such an estimator and justify it in a fairly rigorous manner.

Part I

Undisclosed Portion [Withheld]

Part II

Disclosed Portion

2 Plain English Explanation of the Problem

2.1 Timing Metadata in Monero Transactions

Through great effort, Monero researchers and software developers have been able to conceal most information about transactions even though all Monero nodes share the blockchain database itself. However, transactions still contain some information that distinguishes them from one another.

One key piece of information embedded in each Monero transaction is the time that it was confirmed in a mined block. Timestamps are an inherent property of blockchain-based cryptocurrencies. The timestamp ensures that coins are not double-spent.

Each Monero transaction creates at least two *outputs*, which are amounts of Monero that can be spent in a subsequent transaction. When a user that has received an output wishes to spend it, the wallet software he or she uses will include that output in the set of outputs within the ring signature. Monero's current ring size is 16, so there will be 15 *decoy* outputs listed in the ring and 1 *real spend* output.

Table 1: Ring Members of Transaction `8bc3b73e48881d48cd69f1bf82a06b6a7b94bfd0dbf85edac535dab9270a305b`

Output Public Key	Block Number	Block Timestamp
`033a3d4b817bfaba6472f249fdb0dc6b00ecfe20983cea5e608fe5e679b0d23b`	2666408	2022-07-13 13:27:42
`866c4cf39af3e8785cbda21a9cbaa120424df76d5d05518d3ec888b94acf4c2b`	2683943	2022-08-06 20:03:35
`c8c3f8660776fad9dee0dd9bd6a3e02f52ba5e4b4785419db7a4131f1d1aa1ff`	2692667	2022-08-19 02:10:18
`ca2f709c371c21a8ab33cddab4b593c24dd52b0ca9e4835ddcbcf17f0b38e8c7`	2696371	2022-08-24 06:06:50
`1f11eb618ad7bb9a26a436af1190532c3744d0c0c3b335b6d3fd65b0a9911585`	2697585	2022-08-25 21:24:57
`6cd9e83b9f75e44a00310aec8935954a23bcdfa34c4fb5aa5e177212fd45ce95`	2698303	2022-08-26 22:06:16
`9ab87cffbc8eb76a0affbc9d004967ba1ca1ab008ebe68444d687a5d5a1cb145`	2698869	2022-08-27 15:36:55
`52ae53b9b67316c932db550c386fa7092952d65f9a9d52e261dcb0f6ccba78c0`	2699496	2022-08-28 12:52:01
`19357e641d525fd001945d87829045e3ac6ad829ca92c90b729f152e7dc86b4a`	2699840	2022-08-29 00:34:55
`2bcdc2b0b656cefeced925cbe122ecc0bb3d83bf61dfa0ac5a6c96a586ea0560`	2699843	2022-08-29 00:39:49
`56672f13802cca5363cd5def25c9af4be5ad2e625401d6381568902dca5b0b35`	2700102	2022-08-29 09:31:16
`bbae1b9ca63e017d030e295a5c9f92ed13152c82a32fdc10458912b9e0fc4e1a`	2700241	2022-08-29 14:12:24
`cc94142537e87fb36713c84c2bf8465bde03f442d155391025e078bac2154cef`	2700606	2022-08-30 02:04:45
`dda4c8f4c51b387e03b5ce97b187a5cbb4fbc03ea622c62de5afb698f0351215`	2700755	2022-08-30 06:59:12
`846717c3b56dcf2c87a92b71890fdde13c72081d8d7e40ddbcba2e82d866c063`	2700908	2022-08-30 12:10:55
`6241dc9af536fd23584c11b6a9c84c9fd692fc5dbc9326741db6dde3f05832e2`	2700918	2022-08-30 12:33:33

Source: https://xmrchain.net/tx/8bc3b73e48881d48cd69f1bf82a06b6a7b94bfd0dbf85edac535dab9270a305b

The timestamp of all of those outputs in a ring is available to anyone running a full Monero node or even someone who knows how to use a web-based Monero block explorer. Table 1 contains an example of a ring with the time stamps of each ring member. Timing information is not the only data that is available to external observers of the Monero blockchain, but the timing data presents special challenges. Over a year, there will be about $30 \cdot 24 \cdot 365 = 262,800$ different timestamps (each timestamp corresponding to a unique block on the Monero blockchain). Since the number of timestamp values is so large compared to the number of decoys, timing information can help an anti-privacy adversary possibly narrow down which ring member is the real spend.

Other types of data on the blockchain cannot take so many values in practice. The number of outputs of a transaction are limited to 16. Currently the maximum number of inputs of a transaction is about 146, but in practice

the vast majority of transactions have fewer than 5 inputs.[1] Therefore, the amount of information contained in timestamp data dwarfs the information from the number of inputs or outputs of a transaction. Transaction fee and the `tx_extra` field in theory could have many distinguishing values. In practice, the "official" reference wallet software allows users to set one of only 4 fee priority levels and restricts the format of data in the `tx_extra` field. The difference between the timing data and other types of observable data in a Monero transaction is similar to the difference between data on a person's birth date and race. Birth date reveals much more information than race since it can take so many values. Race can only be a pretty limited set of values. For an in-depth discussion of metadata available to Monero blockchain observers, see [Krawiec-Thayer et al., 2021].

2.2 Decoys Must Be Credible

The core challenge of decoys is intuitive: Like decoys in all contexts, a decoy only serves its purpose if — to observers — it looks like the real thing. Unfortunately, previous versions of Monero did a poor job of selecting decoys that looked like the real thing. According to one estimate, 80 percent of Monero transactions prior to February 2017 could be traced simply by guessing that the youngest ring member was the real spend because selected decoys tended to be much older than the real spends ([Möser et al., 2018]). Monero's decoy selection algorithm has been changed in recent years to correct flaws found by existing research, but there is still a lot of room for improvement.

To explain the problem I will use the extreme case of what can happen if a given real spend has zero decoys nearby in the timestamp distribution. Monero's actual decoy selection algorithm does not suffer from this extreme flaw, but it serves its purpose as an introduction to the issue.

Figure 1: Adequate Decoy Selection Algorithm

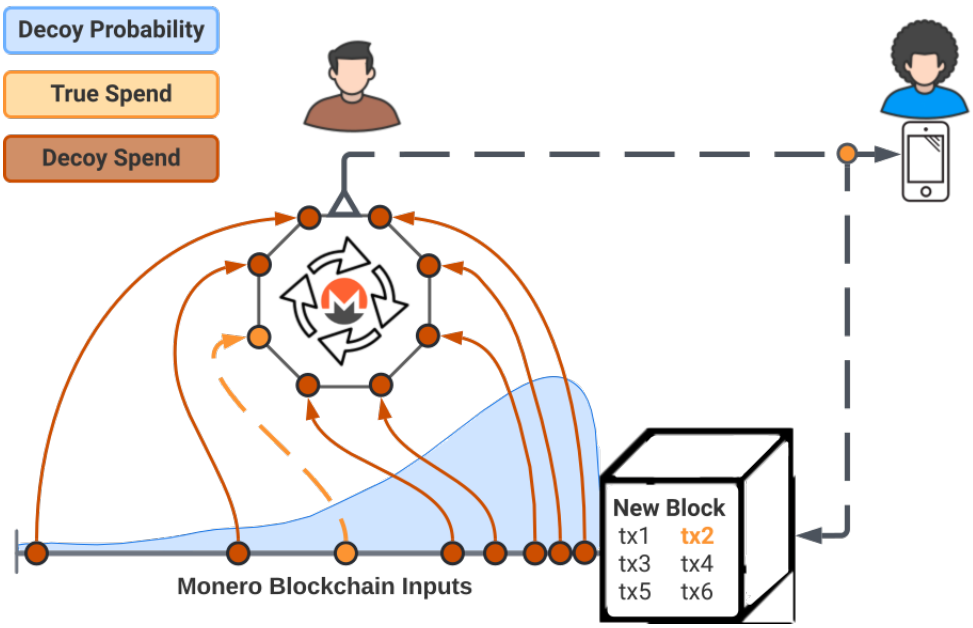


Image designed by ACK-J (https://github.com/ack-j)

The height of the blue shape in Figure 1 represents the probability that the decoy selection algorithm selects outputs as decoys from certain blockchain time stamps. Younger outputs are more likely to be selected as decoys since younger outputs are also more likely to be the real spends. Since there is a high probability that a decoy (red circle) could be selected from the same time interval as the real spend (orange), it is difficult for an anti-privacy adversary to deduce that the orange circle is the real spend.

[1] See discussion: https://libera.monerologs.net/monero-dev/20220526

Figure 2 shows what would happen if the decoy selection algorithm were severely defective. There is a portion of the age distribution that the decoy selection algorithm will never select from. The height of the blue shape is zero at that interval. In this hypothetical, the fact that no decoy would be selected in that interval would be known by an anti-privacy adversary since the decoy selection algorithm is written into Monero's public open source code, at least for Monero wallets that are open source. Therefore, there would be a 100 percent probability that the ring member selected in that interval was the real spend. This particular transaction would be 100 percent traceable. To repeat, this is the extreme case that does not happen in practice. The current decoy selection algorithm is not severely defective like this hypothetical example, but it has significant shortcomings.

Figure 2: Defective Decoy Selection Algorithm

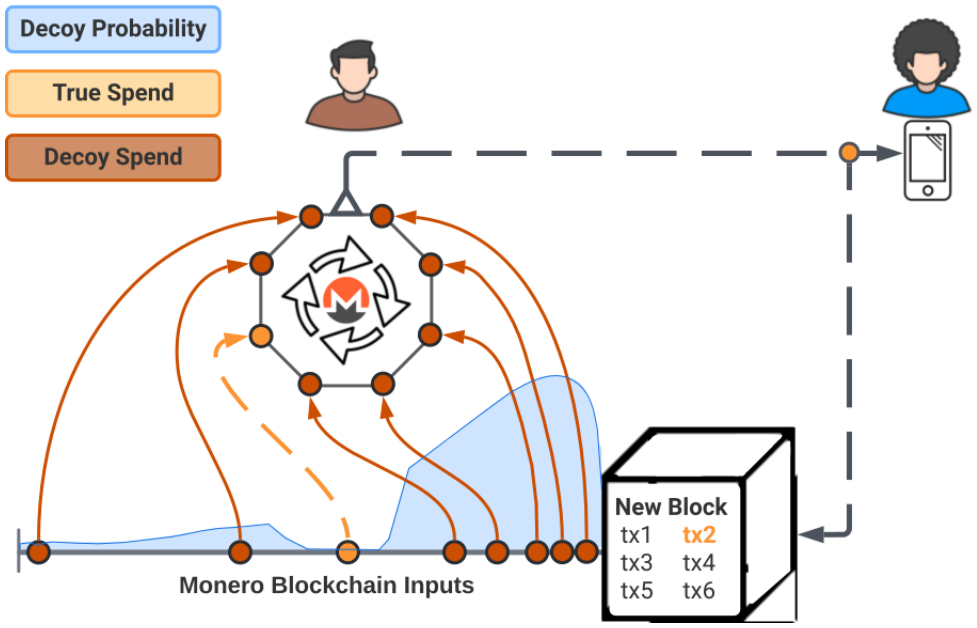


Image designed by ACK-J (https://github.com/ack-j)

The ideal decoy selection algorithm would select decoys from every point on the age distribution in exact proportion to the real spends. For a ring size of 16, this ideal algorithm would provide 15 decoys of "camouflage" on top of every real spend across the entire age distribution. In ideal circumstances, such an algorithm would not give any hint about the real spend to an anti-privacy adversary. The best an adversary could do is random guessing about the real spend, which would only achieve a success rate of $1/16 = 6.25\%$. The "defective" algorithm described above provided zero decoys in a certain age interval, which would lead to full traceability of transactions that spend outputs that were a certain age. Monero's current decoy selection algorithm lies somewhere between the two extremes of ideal and severely defective. The goal of OSPEAD is to move the decoy selection algorithm much closer to the ideal shape.

3 The Solution

The goal of OSPEAD is to use a special type of mathematical function called a parametric probability density function to match the real spend age distribution as closely as possible. The close match will provide about 15 decoys for every real spend along nearly every interval of the age distribution. The match cannot be perfect, however. Parametric probability density functions can only serve as an approximation of complex real human behavior.

There are only a limited number of decoys available per ring. Therefore, if more decoy "camouflage" is moved from section A to section B on the age distribution, users who happen to spend outputs of an age that falls into

section A will experience a reduction in privacy; users spending from section B will experience an increase in privacy. Therefore with the imperfect parametric probability distribution functions there is an unavoidable trade-off. The terms of the trade-off can be made precise mathematically, but the correct choice of trade-off is a judgment call.

Given that the "correct" trade-off is a judgment call, OSPEAD involves a procedure to obtain the best decoy selection algorithm under several different sets of criteria. Once these candidate algorithms are determined, a single best algorithm will be selected though a judgment call. In the next section I describe the proposed sets of criteria: privacy impoverishment, economic welfare, inequality minimization, worst-case-scenario minimization, Maximum Likelihood Estimation, and maximize resistance to a specific attack.

4 Criteria for Best Fit

At first glance constructing $f_D(x)$, the probability density function (PDF) of the decoy selection algorithm, appears to be a standard problem: Extract the real spend age distribution $f_S(x)$ "data" through some method and then fit some distribution to it in order to form $f_D(x)$. Typically in applied statistical work, "fitting some distribution" would be done via nonparametric means or Maximum Likelihood Estimation (MLE) for a parametric approach. Since we are setting aside nonparametric methods in the short term, MLE may see like the way forward.

I would argue instead that what we confront here is not actually a standard problem, despite appearances. An MLE approach, which is the approach taken in [Möser et al., 2018], may make sense if what we are interested in is conducting hypothesis tests on the parameters of parametric distribution families, a typical scientific exercise. In fact, MLE is quite good at that task, assuming certain crucial assumptions are met, since asymptotically it is usually guaranteed to achieve minimum variance among unbiased estimators.

What we are trying to do here with construction of $f_D(x)$ has little to do with conducting hypothesis tests, however. Most statistical methods are interested in separating statistical signal from noise. What we are trying to do is quite the opposite: merge the signal (real spends) and noise (decoys) so that even a determined and sophisticated statistician cannot distinguish them. We are not trying to minimize variance in fitting $f_D(x)$ to $f_S(x)$. Rather, we are trying to minimize risk to user privacy. We are trying to minimize traceability of Monero transactions. This goal requires us to be more creative in our approach.

In general, our goal is to "cover" every age interval with 15 decoys for every real spend that comes from that interval. A user who spends from a point of the age distribution that is not covered by at least 15 decoys experiences a privacy deficit. A user who spends from a point of the age distribution that is covered by more than 15 decoys has a surplus of decoys. To approach the ideal of all users being protected by a single-hop anonymity set of 16, we can move these decoys from the surplus at particular age x values to the x values with deficits. I will build OSPEAD around the idea of minimizing this privacy deficit for all x, defined as $h(x) = \max\{0, f_S(x) - f_D(x)\}$. I also include a symmetric option: $h_{sym}(x) = |f_S(x) - f_D(x)|$ as a possible minimization objective.

To justify my proposed methodology, I will quote extensively from the second edition of *Statistical Inference* by George Casella & Roger L. Berger ([Casella & Berger, 2002]). In the book's preface, the authors write, "The purpose of this book is to build theoretical statistics (as different from mathematical statistics) from the first principles of probability theory....The book is intended for first-year graduate students majoring in statistics or in a field where a statistics concentration is desirable." As far as I can tell, it is widely used for that purpose, at least within economics. For example, the first course in the econometrics sequence of MIT's economics doctoral program uses the book as its primary textbook.[2] Therefore, in my view the book is authoritative enough to lean on in justifying my proposed approach.

I will begin by quoting from pages 348–350:

[2] https://ocw.mit.edu/courses/economics/14-381-statistical-method-in-economics-fall-2018/syllabus/
Other economics doctoral programs likely do as well, but we cannot know for sure since many of them keep their syllabi hidden behind student login screens.

7.3.4 Loss Function Optimality

Our evaluations of point estimators have been based on their mean squared error performance. Mean squared error is a special case of a function called a *loss function*. The study of the performance, and the optimality, of estimators evaluated through loss functions is a branch of *decision theory*.

After the data $\mathbf{X} = \mathbf{x}$ are observed, where $\mathbf{X} \sim f(\mathbf{x}|\theta)$, $\theta \in \Theta$, a decision regarding θ is made. The set of allowable decisions is the *action space*, denoted by $\mathcal{A}$. Often in point estimation problems $\mathcal{A}$ is equal to Θ, the parameter space, but this will change in other problems (such as hypothesis testing—see Section 8.3.5).

The loss function in a point estimation problem reflects the fact that if an action a is close to θ, then the decision a is reasonable and little loss is incurred. If a is far from θ, then a large loss is incurred. The loss function is a nonnegative function that generally increases as the distance between a and θ increases. If θ is real-valued, two commonly used loss functions are

$$\textit{absolute error loss}, \quad L(\theta, a) = |a - \theta|,$$

and

$$\textit{squared error loss}, \quad L(\theta, a) = (a - \theta)^2.$$

Both of these loss functions increase as the distance between θ and a increases, with minimum value $L(\theta, \theta) = 0$. That is, the loss is minimum if the action is correct. Squared error loss gives relatively more penalty for large discrepancies, and absolute error loss gives relatively more penalty for small discrepancies. A variation of squared error loss, one that penalizes overestimation more than underestimation, is

$$L(\theta, a) = \begin{cases} (a - \theta)^2 & \text{if } a < \theta \\ 10(a - \theta)^2 & \text{if } a \geq \theta. \end{cases} \tag{1}$$

A loss that penalizes errors in estimation more if θ is near 0 than if $|\theta|$ is large, a relative squared error loss, is

$$L(\theta, a) = \frac{(a - \theta)^2}{|\theta| + 1}.$$

Notice that both of these last variations of squared error loss could have been based instead on absolute error loss. **In general, the experimenter must consider the consequences of various errors in estimation for different values of θ and specify a loss function that reflects these consequences.** [my emphasis]

In a loss function or *decision theoretic* analysis, the quality of an estimator is quantified in its *risk function*; that is, for an estimator $\delta(\mathbf{x})$ of θ, the risk function, a function of θ, is

$$R(\theta, \delta) = \mathrm{E}_\theta L(\theta, \delta(\mathbf{X})). \tag{2}$$

At a given θ, the risk function is the average loss that will be incurred if the estimator $\delta(\mathbf{x})$ is used.

Since the true value of θ is unknown, we would like to use an estimator that has a small value of $R(\theta, \delta)$ for all values of θ. This would mean that, regardless of the true value of θ, the estimator will have a small expected loss. If the qualities of two different estimators, δ_1 and δ_2, are to be compared, then they will be compared by comparing their risk functions, $R(\theta, \delta_1)$ and $R(\theta, \delta_2)$. If $R(\theta, \delta_1) < R(\theta, \delta_2)$ for all $\theta \in \Theta$, then δ_1 is the preferred estimator because δ_1 performs better for all θ. More typically, the two risk functions will cross. Then the judgment as to which estimator is better may not be so clear-cut.

The risk function for an estimator δ is the expected loss, as defined in (2). For squared error loss, the risk function is a familiar quantity, the mean squared error (MSE) that was used in Section 7.3.1. There the MSE of an estimator was defined as $\mathrm{MSE}(\theta) = \mathrm{E}_\theta(\delta(\mathbf{X}) - \theta)^2$, which is just $\mathrm{E}_\theta L(\theta, \delta(\mathbf{X})) = R(\theta, \delta)$ if $L(\theta, a) = (a - \theta)^2$. As in Chapter 7 we have that, for squared error loss,

$$R(\theta, \delta) = \mathrm{Var}_\theta\, \delta(\mathbf{X}) + (\mathrm{E}_\theta \delta(\mathbf{X}) - \theta)^2 = \mathrm{Var}_\theta\, \delta(\mathbf{X}) + (\mathrm{Bias}_\theta\, \delta(\mathbf{X}))^2.$$

This risk function for squared error loss clearly indicates that a good estimator should have both a small variance and a small bias. A decision theoretic analysis would judge how well an estimator succeeded in simultaneously minimizing these two quantities.

It would be an atypical decision theoretic analysis in which the set $\mathcal{D}$ of allowable estimators was restricted to the set of unbiased estimators, as was done in Section 7.3.2. [my emphasis] Then, minimizing the risk would just be minimizing the variance. A decision theoretic analysis would be more comprehensive in that both the variance and bias are in the risk and will be considered simultaneously. An estimator would be judged good if it had a small, but probably nonzero, bias combined with a small variance.

The text then goes into several specific examples of risk analysis and then briefly explores risk within a Bayesian framework. What does Section 7.3.2 (page 334), referenced above, say?

7.3.2 Best Unbiased Estimators

As noted in the previous section, a comparison of estimators based on MSE considerations may not yield a clear favorite. Indeed, there is no one "best MSE" estimator. Many find this troublesome or annoying, and rather than doing MSE comparisons of candidate estimators, they would rather have a "recommended" one.

The reason that there is no one "best MSE" estimator is that the class of all estimators is too large a class. (For example, the estimator $\hat{\theta} = 17$ cannot be beaten in MSE at $\theta = 17$ but is a terrible estimator otherwise.) One way to make the problem of finding a "best" estimator tractable is to limit the class of estimators. A popular way of restricting the class of estimators, the one we consider in this section, is to consider only unbiased estimators.

If W_1 and W_2 are both unbiased estimators of a parameter θ, that is, $\mathrm{E}_\theta W_1 = \mathrm{E}_\theta W_2 = \theta$, then their mean squared error are equal to their variances, so we should choose the estimator with the smaller variance. If we can find an unbiased estimator with uniformly smallest variance—a best unbiased estimator—then our task is done.

[Casella & Berger, 2002] then go on to give a formal definition of best unbiased estimator — also known as uniform minimum variance unbiased estimator (UMVUE) — in Definition 7.3.7 and then give the Cramér–Rao Lower Bound (CRLB) in Theorem 7.3.9 and Corollary 7.3.10. Corollary 7.3.15 is also useful. The gist of the discussion is that certain unbiased estimators of a parameter θ (the text uses the notation $\tau(\theta)$, with τ being some continuous function of θ, in order to be more general) of a distribution $f(\mathbf{x}|\theta)$ may achieve the lowest variance possible in this setting: the Cramér–Rao Lower Bound (CRLB).

Now I will move on to examining Maximum Likelihood Estimation (MLE) in this entire context. First I will reference the definition of asymptotic variance from [Casella & Berger, 2002], page 471. Let k_n be some normalizing constant.

Definition 10.1.9 For an estimator T_n, suppose that $k_n(T_n - \tau(\theta)) \to n(0, \sigma^2)$ in distribution. The parameter σ^2 is called the *asymptotic variance* or *variance of the limit distribution* of T_n.

I also need their definition of asymptotic efficiency to make my point:

Definition 10.1.11 A sequence of estimators W_n is asymptotically efficient for a parameter $\tau(\theta)$ if $\sqrt{n}\,[W_n - \tau(\theta)] \to n\,[0, v(\theta)]$ in distribution and

$$v(\theta) = \frac{[\tau'(\theta)]^2}{\mathrm{E}_\theta\left(\left(\frac{\partial}{\partial\theta}\log f\,(X|\theta)\right)^2\right)};$$

that is, the asymptotic variance of W_n achieves the Cramér–Rao Lower Bound.

Finally, we come to one of the primary reasons MLE is so popular in applied statistical work:

> **Theorem 10.1.12 (Asymptotic efficiency of MLEs)** Let $X_1, X_2, ...,$ be iid $f(x|\theta)$, let $\hat{\theta}$ denote the MLE of θ, and let $\tau(\theta)$ be a continuous function of θ. Under the regularity conditions in Miscellanea 10.6.2 on $f(x|\theta)$ and, hence $L(\theta, \mathbf{x})$ [this expression, the likelihood function, is defined in Theorem 10.1.6 as $L(\theta|\mathbf{x}) = \prod_{i=1}^{n} f(x_i|\theta)$],
>
> $$\sqrt{n}\left[\tau(\hat{\theta}) - \tau(\theta)\right] \rightarrow n\left[0, v(\theta)\right],$$
>
> where $v(\theta)$ is the Cramér–Rao Lower Bound. That is, $\tau(\hat{\theta})$ is a consistent and asymptotically efficient estimator of $\tau(\theta)$.

Later, at the top of page 477, [Casella & Berger, 2002] make this comment:

> Since the MLE is typically asymptotically efficient, another estimator cannot hope to beat its asymptotic variance. However, other estimators may have other desirable properties (ease of calculation, **robustness to underlying assumptions** [my emphasis]) that make them desirable. In such situations, the efficiency of the MLE becomes important in calibrating what we are giving up if we use an alternative estimator.

Let us break down what [Casella & Berger, 2002] are saying in these passages, and how they relate to OSPEAD. They set up a framework for decision-making based on statistical analysis. Loss functions, decision theory, and risk functions are the main elements of this framework. They then discuss some of the most common loss functions, Mean Squared Error (MSE) being one of them. I note in passing that they discuss a "variation" on MSE in equation (1) that bears some resemblance to my privacy deficit notion, defined as $h(x) = \max\{0, f_S(x) - f_D(x)\}$, in that it is asymmetric in penalization of under- and over-estimation of some quantity.

Minimizing MSE for estimates of a particular parameter θ is a fair goal for statistical analyses whose purpose is to conduct traditional hypothesis tests regarding the true value of θ in line with the Popperian falsification paradigm of science. Lower MSE generally would lead to higher statistical power and therefore greater ability to avoid Type II error in hypothesis tests. [Casella & Berger, 2002] write, "In general, the experimenter must consider the consequences of various errors in estimation for different values of θ and specify a loss function that reflects these consequences." By minimizing bias and variance explicitly, MSE — as a loss function — performs well in avoiding the Type I and Type II error consequences in null hypothesis testing and therefore MSE often makes sense to use as a loss function in scientific hypothesis testing.

[Casella & Berger, 2002] go further and observe, "It would be an atypical decision theoretic analysis in which the set $\mathcal{D}$ of allowable estimators was restricted to the set of unbiased estimators, as was done in Section 7.3.2." As I highlighted, MLE typically has lowest variance, asymptotically, among estimators with zero bias. Therefore, restricting ourselves to MLE in tackling the problem before us — construction of the PDF for the decoy selection algorithm — would likely seem ill-advised to [Casella & Berger, 2002], since the class of unbiased estimators might not be suitable.

Note also that maximum likelihood estimators are point estimators for the particular θ parameters of PDFs. In theory, the estimated $\hat{\theta}$ will yield the corresponding theoretical distributions that fit the target empirical distributions well, although only under the assumption that the empirical distribution being fitted exactly equals the chosen theoretical PDF. There is no guarantee that variance — or bias for that matter — for the distribution itself will be small when using MLE to fit the wrong theoretical PDF to an empirical distribution. Given the fact that we are interested in the whole distributions themselves, i.e. $f_S(x)$ and $f_D(x)$, and not just parameters of parametric

distributions, the justification for using MLE in this setting is even weaker. In just a moment I will convert the problem of fitting distributions into a parametric one, so that the true problem is better illustrated.

Until now I have dealt with $f_S(x)$ and $f_D(x)$ as if they were probability density functions, i.e. as if the domain of the functions were continuous. However, in reality they are probability mass functions, i.e. the domain of the functions are discrete. The real spend age distributions are only meaningful in terms of the discrete blocks on the blockchain, since miners can arrange valid transactions within a block in any order they choose. We can leverage this fact for the following analysis.

Define a set of parameters $\boldsymbol{\theta}$ as follows:

$$\boldsymbol{\theta} = \{\theta_1, \theta_2, ..., \theta_N\} = \{f_S(1), f_S(2), ..., f_S(N)\}$$

So θ_1 is the value of $f_S(x)$ when $x = 1$, i.e. the first-available block that an output can be spent in. θ_N refers to the first block where RingCT outputs appeared. My definition here is not completely rigorous, since the blockchain is lengthening constantly, and therefore the number of elements of $\boldsymbol{\theta}$ is increasing by the hour.

What we wish to estimate is $\boldsymbol{\theta}$ — every element of it. Or, more specifically, we wish to construct some $f_D(x)$ such that $f_D(x_i)$ is "close" — in some sense to be defined shortly — to each θ_i, for every i. With a nonparametric approach, we may be able to tackle each element θ_i individually, more or less. With a parametric approach, we cannot hope to do so. Therefore, we must establish some overall metric, or metrics, of "success". In other words, we must define and justify a set $\mathcal{L}$ of loss functions. This is our first task. Our second task is to define a set $\mathcal{D}$ of allowable estimators.

In equation (2), [Casella & Berger, 2002] defined the risk function as $R(\theta, \delta) = \mathrm{E}_\theta L(\theta, \delta(\mathbf{X}))$. For the time being, we will assume that $\boldsymbol{\theta}$ is deterministic and therefore the risk function equals the loss function. The $\boldsymbol{\theta}$ is actually stochastic — it changes over time as user spending patterns change over time — which will be dealt with in **Section 7 Dynamic Risk and Forecasting**.

There are some similarities between the Differential Privacy framework and Monero's ring signature privacy model. Differential Privacy seeks a balance between the desire to perform statistical analysis on private data and the need to protect individuals from discovery of their private information. Monero seeks to maximize privacy of users, but is constrained by reasonable limits on ring size. I looked through the Differential Privacy literature for some criteria for fitting a decoy selection algorithm. In general there are utility-based criteria for resolving the balance between statistical analysis and user privacy for each user, but there is not yet a way to resolve our problem([Hsu et al., 2014]). Our problem is different in that a particular ring size is fixed, and then trade-offs *between users* need to be determined.

In "Research Roadmap for an Overhaul of Monero's Mixin Selection Algorithm", which I submitted to Monero's Vulnerability Response Process in 2021, I wrote:

> How do we construct $f_M(x)$ [the decoy selection algorithm] for best overall privacy if we restrict ourselves to parametric distributions, and therefore cannot achieve $f_S(x) \approx f_M(x)$ for all values of x? Well, it depend [sic] on how we define "best". Below are six approaches. I have the mathematical definitions of these worked out in my head, but they are not written here:
>
> 1) Privacy impoverishment
> 2) Economic welfare
> 3) Inequality minimization
> 4) Worst-case-scenario minimization
> 5) Maximum Likelihood Estimation

6) Maximize resistance to a specific attack

Now that a set of objective functions have been defined, one could imagine optimizing these objective functions in a numerical optimization procedure where each parametric distribution family with support of $[0, \infty)$ is permuted over 1-6 above.

These objective functions are the loss functions in the framework of [Casella & Berger, 2002]. Many of them are forms of minimum divergence estimators ([Maji et al., 2019]). Just two more bits of housekeeping before I can mathematically define these 1-6 loss functions. First, to be consistent with the notation of [Casella & Berger, 2002], define $f_D(x_i) \equiv a_i$.

Second, we must think about how to give weight to each θ_i. Do we give equal weight to every θ_i? That would mean that outputs years old would be given the same importance as outputs just a few minutes or hours old. Weighting the loss function by the value of θ_i seems more reasonable since that would, in effect, give each spent output equal importance. However, we are trying to protect the privacy of people, not outputs.

Say that User A re-spends outputs frequently, and so has real spends that are in the thick portion of $f_S(x)$. Say that User B re-spends outputs infrequently, and therefore has real spends that are in the thin portion of $f_S(x)$. Furthermore, User A would be able to generate more transactions than User B simply because of rapidly re-spending outputs. Therefore, only weighting the loss function by the value of θ_i would (maybe unfairly) give more importance to User A than User B, just due to the way that the mathematics work out. The way forward is not exactly clear. I think it could make sense to try several intermediate weighting schemes.

First, my recommendation is to only give weight to elements of $\boldsymbol{\theta}$ when there is a privacy deficit, i.e. when $h(x) = \max\{0, f_S(x) - f_D(x)\}$ is nonzero. Later we will use $\mathbf{1}\{x\}$, the indicator function, for this part of the weighting scheme, or the $\min\{x\}$ operator, depending on the context.[3] For comparison purposes, I will also include the symmetric counterpart $h_{sym}(x) = |f_S(x) - f_D(x)|$ that seeks to avoid privacy surpluses as much as it seeks to avoid privacy deficits. To allow for multiple intermediate weighting schemes, define this weight function:

$$w(\theta_i, \lambda) = \lambda\theta_i + (1 - \lambda)\tfrac{1}{N} \tag{3}$$

Thus, when $\lambda = 1$, $w(\theta_i, \lambda)$ is fully weighted by the value of θ_i. When $\lambda = 0$, $w(\theta_i, \lambda)$ gives equal weight to each θ_i. Tentatively, let $\boldsymbol{\lambda} = \{0, 0.5, 0.9, 0.95, 0.99, 0.999, 0.9999, 1\}$.

Now we are ready to define the set $\mathcal{L}$ of loss functions.

4.1 Privacy impoverishment

To me, the privacy deficit formulation is somewhat reminiscent of a poverty line. Below a certain defined threshold individuals are considered to be impoverished. In the case of a standard poverty indicator, there is some poverty line z defined by a researcher or government entity. In the case of Monero user privacy, the corresponding "poverty line" would be $f_S(x)$, which is different for every x.

The Foster–Greer–Thorbecke (FGT) indices are a family of widely-used poverty indicators. They are defined as

$$FGT_\alpha = \frac{1}{N}\sum_{i=1}^{H}\left(\frac{z - y_i}{z}\right)^\alpha$$

where N is the number of people in the population under study, H is the number of people within that population who are below the poverty line, z is the poverty line, y_i is the income (or consumption) of each individual i who is under the poverty line, and α is a parameter that controls weighting.

[3] https://en.wikipedia.org/wiki/Indicator_function

If α is high, then people far below the poverty line are given much more weight than people just barely below the poverty line. When $\alpha = 0$, FGT_0 is simply the poverty headcount. When $\alpha = 1$, FGT_1 is the poverty gap index. When $\alpha = 2$, FGT_2 weights each person's poverty gap by its square, and for $\alpha = 3$, by its cube, and so forth.

A loss function analogue can be defined:

$$L_{FGT_\alpha}(\boldsymbol{\theta}, \boldsymbol{a}, \lambda) = \frac{1}{N} \sum_{i=1}^{N} \mathbf{1}\left\{a_i < \theta_i\right\} w(\theta_i, \lambda) \left(\frac{\theta_i - a_i}{\theta_i}\right)^{\alpha} \tag{4}$$

The $\mathbf{1}\left\{a_i < \theta_i\right\}$ indicator function ensures that a_i's are only counted when there is some privacy deficit. The symmetric counterpart $L_{FGT_\alpha,sym}(\boldsymbol{\theta}, \boldsymbol{a}, \lambda)$ would omit the $\mathbf{1}\left\{a_i < \theta_i\right\}$ term and replace $\left(\frac{\theta_i - a_i}{\theta_i}\right)^{\alpha}$ with $\left(\left|\frac{\theta_i - a_i}{\theta_i}\right|\right)^{\alpha}$. Using $\alpha = 0$ probably doesn't make sense for our purposes. Using $\boldsymbol{\alpha} = \{1, 2, 3\}$ is reasonable.

4.2 Economic welfare

A full discussion of the meaning and theory of economic welfare is outside of the scope of this document. The basic idea is that people are theorized to have preferences that can be approximated via a mathematical function. These approximating functions are called utility functions. When people's preferences are satisfied, they obtain utility. When the preferences are satisfied to a higher degree, they obtain higher utility. In essence, when an individual's utility is higher, their happiness is higher. Within the framework of neoclassical economics, individuals strive to maximize their utility, subject to the constraints they encounter in the world. The analysis of aggregate population-level utility is the realm of welfare economics.

We may assume that Monero users prefer to have privacy and therefore some "privacy utility function" in this specific context could be defined. In practice, the loss function derived from this type of analysis will look similar to the $L_{FGT_\alpha}(\boldsymbol{\theta}, \boldsymbol{a}, \lambda)$ privacy impoverishment loss function, but with a different interpretation.

There are dozens of utility functions to choose from. One of the more appropriate ones in the present context is the Constant Relative Risk Aversion (CRRA) utility function:

$$u_{CRRA_\eta}(y) = \begin{cases} \frac{y^{1-\eta}}{1-\eta} & \eta \geq 0, \eta \neq 1 \\ \ln(y) & \eta = 1 \end{cases} \tag{5}$$

where η is the coefficient of constant relative risk aversion. The CRRA utility function makes sense to use in this context because (1) At a basic level, it takes a single argument, unlike other classes of utility functions that deal with multiple goods and services; (2) It explicitly deals with risk; (3) It has an adjustable parameter η that we can use to explore the sensitivity of the results; (4) For $\eta \geq 1$, the CRRA utility function approaches $-\infty$ as y approaches zero.

The (4) characteristic serves as an important advantage compared to the privacy impoverishment framework since in theory the $L_{FGT_\alpha}(\boldsymbol{\theta}, \boldsymbol{a}, \lambda)$ loss function would allow $f_D(x)$ to have zero mass at some x values — and therefore any ring members having an age corresponding to those x values would be clearly identifiable as real spends. In addition, a numerical optimization process that used the CRRA utility function as its basis would avoid at all costs $f_D(x) = 0$ for any and all x values as long as η is chosen so that $\eta \geq 1$.

In a welfare economics framework, generally some discussion of the Pareto weights and Pareto efficiency would be appropriate. However, that type of discussion has little practical effect on the analysis here and it would generally only be of interest to economists, so I will elide it here.

Now we are ready to define an economic welfare loss function:

$$L_{Welfare_\eta}(\boldsymbol{\theta}, \boldsymbol{a}, \lambda) = (-1)\,\frac{1}{N}\sum_{i=1}^{N} w(\theta_i, \lambda)\left(u_{CRRA_\eta}\left(\min\left\{\frac{a_i}{\theta_i}, 1\right\}\right)\right) \tag{6}$$

The (-1) scalar is to make this a *loss* function to be minimized. As is typical, the CRRA utility function increases with its argument, and privacy increases as a_i becomes closer to θ_i, i.e. as $f_D(i)$ becomes closer to $f_S(i)$. We use the $\min\{x\}$ operator here. When $a_i < \theta_i$, the utility function is applied to $\frac{a_i}{\theta_i}$. When $a_i \geq \theta_i$, the utility function is applied to one, as if $a_i = \theta_i$, since users spending outputs from this block i suffer no privacy deficit as I have defined it. The symmetric counterpart $L_{Welfare_\eta,sym}(\boldsymbol{\theta}, \boldsymbol{a}, \lambda)$ would replace $\min\left\{\frac{a_i}{\theta_i}, 1\right\}$ with $\frac{a_i}{\theta_i}$.

The choice for η is somewhat open-ended. Perhaps five values should be tested, to get some sense of the sensitivity of the results to the choice of η. One of the values should be $\eta = 1$ so that log utility is used. It is less clear what the remaining four values should be. Some experimentally-determined values for η for the utility of income are available in the literature, but how individuals' utility functions for income and privacy relate to one another is unclear. Tentatively, let us set $\boldsymbol{\eta}$ to $\boldsymbol{\eta} = \{0.5, 1, 2, 5\}$.

4.3 Inequality minimization

Another possible loss function is an inequality metric. The privacy impoverishment and economic welfare frameworks dealt with the absolute privacy of each user. We may also want to consider a framework that explicitly compares users to each other, in terms of the privacy that a $f_D(x)$ provides. A loss function that attempts to minimize inequality of privacy among users may be appropriate.

One of the most widely-used inequality metrics is the Gini coefficient (or Gini index). The Gini coefficient has many attractive theoretical properties that I will not recite here. Its main drawback is that its value is not very interpretable for laypeople. (The most intuitive explanation involves the Lorenz curve.) Given that we are already far into the realm of difficulty with interpretation of these loss functions, the low interpretability of the Gini coefficient should not stop us from using it.

One formulation of the Gini coefficient is:

$$G(\mathbf{x}) = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n}|x_i - x_j|}{2\sum_{i=1}^{n}\sum_{j=1}^{n} x_j} = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n}|x_i - x_j|}{2n^2\bar{x}}$$

This formulation is somewhat easy to interpret, although it is computationally expensive as it requires about $\left(n^2 - n\right)/2$ arithmetic operations. It does not contemplate weights, so it must be modified. [Creedy, 2015] provides some guidance.

First, the formula suggested by Creedy requires that the weights sum up to n, i.e. $\sum_{i=1}^{n} w_i = n$. The weights that we will use, $w(\theta_i, \lambda)$ could easily be normalized to ensure $\sum_{i=1}^{n} w(\theta_i, \lambda) = n$.

In his equation (17), Creedy suggests the following weighted formula:

$$G(\mathbf{x}) = \frac{\sum_{i=1}^{n}\sum_{j=i+1}^{n} w_i w_j\,|x_i - x_j|}{\sum_{i=1}^{n} w_i \sum_{i=1}^{n} w_i x_i}$$

If we let x_i be $\frac{a_i}{\theta_i}$ as with the economic welfare loss function, the corresponding loss function for our problem would then be:

$$L_{Gini}(\boldsymbol{\theta}, \boldsymbol{a}, \lambda) = \frac{\sum_{i=1}^{n}\sum_{j=i+1}^{n} w(\theta_i, \lambda) \cdot w(\theta_j, a_j, \lambda) \cdot \left|\min\left\{\frac{a_i}{\theta_i}, 1\right\} - \min\left\{\frac{a_j}{\theta_j}, 1\right\}\right|}{\sum_{i=1}^{n} w(\theta_i, \lambda) \sum_{i=1}^{n} w(\theta_i, \lambda) \cdot \min\left\{\frac{a_i}{\theta_i}, 1\right\}} \tag{7}$$

$$\text{subject to} \quad \textstyle\sum_{i=1}^{n} w(\theta_i, \lambda) = n$$

Note that the formula given in equation (17) in [Creedy, 2015] is just one possible way to deal with weighting to compute a Gini coefficient. The reason that there are multiple ways to do weighting is that weighting attempts to, in essence, interpolate parts of the Lorenz curve. How to do that interpolation is up for debate. In his equation (12), [Creedy, 2015] also gives a weighted Gini formula that is less computationally expensive. However, it would require conversion of $w(\theta_i, \lambda)$ to integers, which is feasible but adds another layer of complication. The symmetric counterpart $L_{Gini,sym}(\boldsymbol{\theta}, \boldsymbol{a}, \lambda)$ would replace $\min\left\{\frac{a_i}{\theta_i}, 1\right\}$ with $\frac{a_i}{\theta_i}$.

4.4 Worst-case-scenario minimization

Worst-case-scenario minimization is inspired by Appendix F: Minimum Untraceability in [Möser et al., 2018]. They define Ge_{min}, the minimum possible guessing entropy of the untraceability of a transaction input:

$$\text{Ge}_{min} = \frac{\frac{1}{2}m(m+1)}{\frac{r_{max}}{r_{min}} + m} \tag{8}$$

where m is the number of ring members and

$$r_{max} = \max_{\forall x}\left(\frac{f_S(x)}{f_D(x)}\right), r_{min} = \min_{\forall x}\left(\frac{f_S(x)}{f_D(x)}\right)$$

Since m is a constant, and optimization procedures are insensitive to constants, for the purposes of constructing a loss function, Ge_{min} can be simplified to

$$\text{Ge}_{min\ (optimizer)} = \frac{r_{max}}{r_{min}} \tag{9}$$

The logic to construct $\text{Ge}_{min\ (optimizer)}$ is as follows:

$$\begin{aligned}
\text{Ge}_{min\ (optimizer)} &= \frac{\frac{1}{2}m(m+1)}{\frac{r_{max}}{r_{min}} + m} \\
&\propto \frac{1}{\frac{r_{max}}{r_{min}} + m} && \text{scaling by } \left(\tfrac{1}{2}m(m+1)\right)^{-1} \text{ is a monotonic transformation} \\
&\cong \frac{r_{max}}{r_{min}} + m && g(x) = \tfrac{1}{x} \text{ is a strictly decreasing monotonic transformation} \\
&\cong \frac{r_{max}}{r_{min}} && \text{subtracting m is a monotonic transformation}
\end{aligned}$$

(I am using the $\cong$ symbol loosely here.) Therefore, a $f_D(x)$ that minimizes (9) will also maximize (8)

Putting these ideas into the loss function notation that I have been using, i.e.

$$r_{max} = \max_{\forall i}\left(\frac{\theta_i}{a_i}\right), r_{min} = \min_{\forall i}\left(\frac{\theta_i}{a_i}\right)$$

gives us

$$L_{Worst\,case}(\boldsymbol{\theta}, \boldsymbol{a}) = \frac{\max\limits_{i\in\{1,\ldots,N\}} \left(\frac{\theta_i}{a_i}\right)}{\min\limits_{i\in\{1,\ldots,N\}} \left(\frac{\theta_i}{a_i}\right)} \tag{10}$$

It is not clear to me how weighting by $w(\theta_i, \lambda)$ might work here since the numerator and denominator are both single numbers. For the time being I will leave it out.

Although I include $L_{Worst\,case}$ in the proposed set $\mathcal{L}$ of loss functions, I am skeptical of its usefulness. It is not clear to me that we should care about the minimum θ_i and a_i among literally hundreds of thousands of them. Maybe the tail should not wag the dog. However, it could serve as a useful comparison to other approaches, so I include it in $\mathcal{L}$. Another issue is that I suspect that $L_{Worst\,case}$ will not be a well-behaved function for the purposes of numerical optimization. Given the min and max operators and how they are used here, I foresee that $L_{Worst\,case}$ as a function will have some significant discontinuities with respect to i.

4.5 Maximum Likelihood Estimation

As I have stated, I do not favor a MLE approach here. However, including it in the set of loss functions $\mathcal{L}$ might serve as a useful comparison.

Let $\boldsymbol{\beta} = \{\beta_1, ..., \beta_k\}$ be the set of parameters of some parametric probability density function $g(x|\boldsymbol{\beta})$. Then the likelihood function for some sample $\mathbf{x}$ is

$$L\left(\boldsymbol{\beta}|\mathbf{x}\right) = \prod_{i=1}^{n} g\left(x_i|\boldsymbol{\beta}\right)$$

To put MLE in a loss function framework, convert it into a minimization problem and take the log to ease the computational burden:

$$L_{MLE}\left(\boldsymbol{\beta}|\mathbf{x}\right) = (-1) \cdot \sum_{i=1}^{n} \log g\left(x_i|\boldsymbol{\beta}\right) \tag{11}$$

4.6 Maximize Resistance to an Attack

Define the potency of an attack on the untraceability of Monero transactions for a specified $f_D(x)$ as $\mathscr{P}\left(f_D(x)\right)$, the unconditional probability of correctly guessing the real spend. Then the corresponding loss function can be defined as

$$L_{Attack}\left(f_D(x)\right) = \mathscr{P}\left(f_D(x)\right) \tag{12}$$

Now that the set $\mathcal{L}$ of loss functions has been defined, the set $\mathcal{D}$ of allowable estimators will be defined. Continuing with the θ_i notation, the allowable estimators will be selected from the set of parametric probability density functions (PDF) $f\left(x|\boldsymbol{\beta}\right)$ such that each θ_i shall be estimated by the image of i under $f\left(x|\boldsymbol{\beta}\right)$. In other words, $\hat{\theta}_i = f\left(i|\boldsymbol{\beta}\right)$ for all i.

Strictly speaking, the estimator here is the minimizer of a specified loss function in the set $\mathcal{L}$ where $a_i = f(i)$ for all i for some specified parametric PDF $f\left(x|\boldsymbol{\beta}\right)$. The set of all PDFs (and probability mass functions (PMFs), for that matter) can be all PDFs whose support is the set $[0, \infty)$. The Wikipedia page on probability distributions

lists over 40 distributions that have such a support.[4] So let us say that the target number of elements of the set of PDFs used in estimation is 40. Note that many distributions are special cases of more general distributions and therefore the actual number of distributions to fit may be lower than 40 in practice. Define the set $\mathcal{F}$ of these PDFs. Let $\mathcal{B}$ be the set of parameters of each $f(x|\boldsymbol{\beta})$ under consideration.

4.7 Formalized Optimization Criteria

First, assume we have a good estimate of $f_S(x)$. Then OSPEAD is the set of procedures that perform the following minimizations:

$$
\begin{array}{ll}
\forall L \in \mathcal{L}, & \text{Loss functions} \\
\forall f(x|\boldsymbol{\beta}) \in \mathcal{F}, & \text{Parametric distributions} \\
\forall \lambda \in \boldsymbol{\lambda}, & \text{Weight parameters for } w(\theta_i, \lambda) \\
\forall \alpha \in \boldsymbol{\alpha}, & \text{Exponents for FGT poverty indicator} \\
\forall \eta \in \boldsymbol{\eta}, & \text{Parameters for CRRA utility} \\
 & \\
\min_{\boldsymbol{\beta} \in \mathcal{B}} L\left(f\left(x|\boldsymbol{\beta}\right)\right) & \text{Numerical optimization problems}
\end{array}
\tag{13}
$$

where

$$
\begin{array}{ll}
\mathcal{L} = \{L_{FGT_\alpha}, L_{Welfare_\eta}, L_{Gini}, L_{Worst\,case}, L_{MLE}, L_{Attack}\} & \text{Equations (4), (6), (7), (10), (11), (12)} \\
\mathcal{F} = \{\text{Roughly 40 PDF/PMFs that have support } [0,\infty)\} & \\
\boldsymbol{\lambda} = \{0, 0.5, 0.9, 0.95, 0.99, 0.999, 0.9999, 1\} & \\
\boldsymbol{\alpha} = \{1, 2, 3\} & \\
\boldsymbol{\eta} = \{0.5, 1, 5, 10, 50\} & \\
\mathcal{B} = \{\text{Sets of parameters associated with each element of } \mathcal{F}\} &
\end{array}
$$

For each loss function there will likely be a unique minimizer $f^*(x|\boldsymbol{\beta})$. However, it is unlikely that every loss function will have the same minimizer. Therefore, determining which candidate $f(x|\boldsymbol{\beta})$ is "best" will be the result of a judgment call, taking into account the totality of evidence and theory.

Why stop with single PDFs when we could also examine mixture distributions? It would be useful to test some mixture distributions composed of two or three different parametric PDFs. The main difficulty is the combinatorics: The number of two-PDF mixture distributions would be $\binom{40}{2} = 780$. Given the loss functions and their variations, that would amount to about 58,500 numerical minimization procedures. There are already practical difficulties with carrying out the minimizations with no mixture distributions. Starting values have to be defined and robust minimization algorithms must be chosen and checked for problems. Even with no mixture distributions, the number of minimizations that I set out to perform, as stated in (13), is about 3,000.[5]

Therefore, it could make sense to choose a small set of the most promising or complementary PDFs to do a second round of OSPEAD with mixture distributions. Another possible enhancement could be incorporating some periodic component in a mixture distribution to account for users' sleep-wake cycle.

[4] https://en.wikipedia.org/wiki/List_of_probability_distributions#Supported_on_semi-infinite_intervals,_usually_[0,%E2%88%9E)

[5] Let $|\mathbf{S}|$ denote the cardinality of set $\mathbf{S}$. Note that $|\boldsymbol{\lambda}| = 8$, $|\boldsymbol{\alpha}| = 3$, $|\boldsymbol{\eta}| = 5$, and $|\mathcal{F}| = 40$. Taking into account the different flavors of the loss functions, $|L_{FGT_\alpha}(\boldsymbol{\theta}, \boldsymbol{a}, \lambda)| = |\boldsymbol{\lambda}| \cdot |\boldsymbol{\alpha}| = 24$, $\left|L_{Welfare_\eta}(\boldsymbol{\theta}, \boldsymbol{a}, \lambda)\right| = |\boldsymbol{\lambda}| \cdot |\boldsymbol{\eta}| = 40$, $|L_{Gini}(\boldsymbol{\theta}, \boldsymbol{a}, \lambda)| = |\boldsymbol{\lambda}| = 8$, $|L_{Worst\,case}(\boldsymbol{\theta}, \boldsymbol{a})| = 1$, $|L_{MLE}(\boldsymbol{\beta}|\mathbf{x})| = 1$, and $|L_{RRA}(f_D(x))| = 1$. Therefore, $|\mathcal{L}| = 75$ and hence $|\mathcal{L}| \cdot |\mathcal{F}| = 3,000$.

5 Dry Run with Old Möser et al. (2018) Data

To see how the minimizers in (13) would work it practice, we can perform a dry run on partially de-anonymized Monero spend age data provided by [Möser et al., 2018] from before February 2017. The final version of OSPEAD will use data from about September 2021 to October 2022, but the dry run is useful as a demonstration.

In the dry run I use a subset of the loss function and distribution functions described in (13). For the loss functions I select L_{FGT_α} with $\alpha = \{1, 2\}$, $L_{Welfare_\eta}$ with $\eta = \{0.5, 1\}$, and L_{MLE}. For the distribution function I choose the Log-gamma, Noncentral F, Right-Pareto Log-normal, Generalized Extreme Value, and Generalized Hyperbolic distributions. I also include mixtures of Log-gamma with F, Generalized Extreme Value, and a Laplace Periodic distribution.

Log-gamma: A two-parameter distribution used to fit the spend age data in [Möser et al., 2018]. The current decoy selection algorithm is based on a particular form of the Log-gamma distribution. The Gamma distribution includes the exponential, Erlang, and chi-square distributions as special cases.

Noncentral F: A three-parameter distribution used often in hypothesis testing.

Right-Pareto Log-normal: A three-parameter distribution that is a special case of the Double Pareto-lognormal distribution, which "arises as that of the state of a geometric Brownian motion (GBM), with lognormally distributed initial state, after an exponentially distributed length of time" [Reed & Jorgensen, 2004].

Generalized Extreme Value: A three-parameter distribution that includes the Gumbel, Fréchet and Weibull distributions as special cases.

Generalized Hyperbolic A six-parameter distribution that includes the Normal Inverse Gaussian, Variance Gamma, and Generalized Hyperbolic Student-t distributions as special cases.

Table 2: Performance of Dry Run with Old Möser et al. (2018) Data

Loss function	L_FGT	L_FGT	L_Welfare	L_Welfare	L_MLE
Loss function parameter	1	2	0.5	1	
Log-gamma	0.1138	0.0574	-1.8542	0.1955	7.65e+07
F	0.1095	0.0462	-1.8681	0.1635	7.67e+07
Right-Pareto Log-normal	0.1073	0.0449	-1.8703	0.1604	7.66e+07
Generalized Extreme Value	0.1169	0.0503	-1.8608	0.1766	7.76e+07
Generalized Hyperbolic	0.1100	0.0455	-1.8684	0.1632	7.62e+07
Log-gamma + F mix	0.1081	0.0402	-1.8721	0.1622	
Log-gamma + GEV mix	0.1095	0.0460	-1.8704	0.1547	
Log-gamma + Laplace Periodic	0.112	0.0572	-1.86	0.195	

Note: Values should be compared down columns. Lower values (darker green) indicate better performance. MLE value is Akaike Information Criterion (AIC).

Table 2 contains the values of the minimums for the various loss functions and parametric distributions. These numbers are useful for demonstration purposes, but the final numbers will look very different. We can see that

the Log-gamma distribution tends to compare poorly to the other distributions. The Right-Pareto Log-normal distribution tends to perform better than other pure distributions. The Log-gamma has poor performance and the F has mediocre performance, but when they are combined in a mixture distribution their performance is consistently good.

Table 3 contains the optimized parameter values for the various loss functions and parametric distributions. An important fact to recognize is that a particular set of parameter values that gives good performance according to one loss function may give poor performance for a different loss function. In these exercises we are not comparing a fixed set of parameter values across various loss functions. Rather, we are allowing each distribution's parameters to be bent to minimize each loss function value.

Following Table 3 is a series of plots showing the fitted distributions. The vertical black lines represent the empirical probability mass function. The vertical green lines extending below the horizontal axis represents the empirical cumulative distribution function. 76 percent of the mass of the Möser et al. (2018) data is less than 10,000 blocks old.

The first five plots show the fitted PDFs themselves. To compare the fitted distribution more closely with the empirical data, the second set of five plots shows the ratio of the fitted distributions $f_D(x)$ to the empirical probability mass function $f_S(x)$.

The code for producing these tables and plots is at https://github.com/Rucknium/OSPEAD

Table 3: Optimized Parameter Values of Dry Run with Old Möser et al. (2018) Data

Distribution	Loss fn	Loss fn param	param_1	param_2	param_3
F	L_FGT	1	0.0258	0.471	7.26
F	L_FGT	2	0.0338	0.294	10.3
F	L_Welfare	0.5	0.0289	0.429	8.1
F	L_Welfare	1	0.0333	0.383	9.27
F	L_MLE	0	0.0473	0.685	11.5
Generalized Extreme Value	L_FGT	1	106	421	2.63
Generalized Extreme Value	L_FGT	2	-72.7	3.14e-29	5.74
Generalized Extreme Value	L_Welfare	0.5	-95.9	1.92e-18	3.88
Generalized Extreme Value	L_Welfare	1	-86.6	6.28e-15	4.34
Generalized Extreme Value	L_MLE	0	99.7	246	2.48
Log-gamma	L_FGT	1	6.48	0.894	
Log-gamma	L_FGT	2	5.15	0.639	
Log-gamma	L_Welfare	0.5	6.25	0.852	
Log-gamma	L_Welfare	1	5.79	0.76	
Log-gamma	L_MLE	0	6.62	0.912	
Right-Pareto Log-normal	L_FGT	1	0.235	4.12	1.38
Right-Pareto Log-normal	L_FGT	2	0.133	3.79	1.21
Right-Pareto Log-normal	L_Welfare	0.5	0.209	4.03	1.32
Right-Pareto Log-normal	L_Welfare	1	0.18	3.92	1.25
Right-Pareto Log-normal	L_MLE	0	0.444	4.99	1.83
F Distribution: param_1 is first degree of freedom parameter; param_2 is second degree of freedom parameter; param_3 is non-centrality parameter.					
Generalized Extreme Value Distribution: param_1 is location parameter; param_2 is scale parameter; param_3 is shape parameter.					
Log-gamma Distribution: param_1 is shape parameter; param_2 is rate parameter.					
Right-Pareto Log-normal Distribution: param_1 is shape parameter; param_2 is mean parameter; param_3 is variance parameter.					
Mixture distributions and Generalized Hyperbolic Distribution are omitted from this table.					

Figure 3: Optimized $f_D(x)$ for loss function L_{FGT_α}, $\alpha = 1$

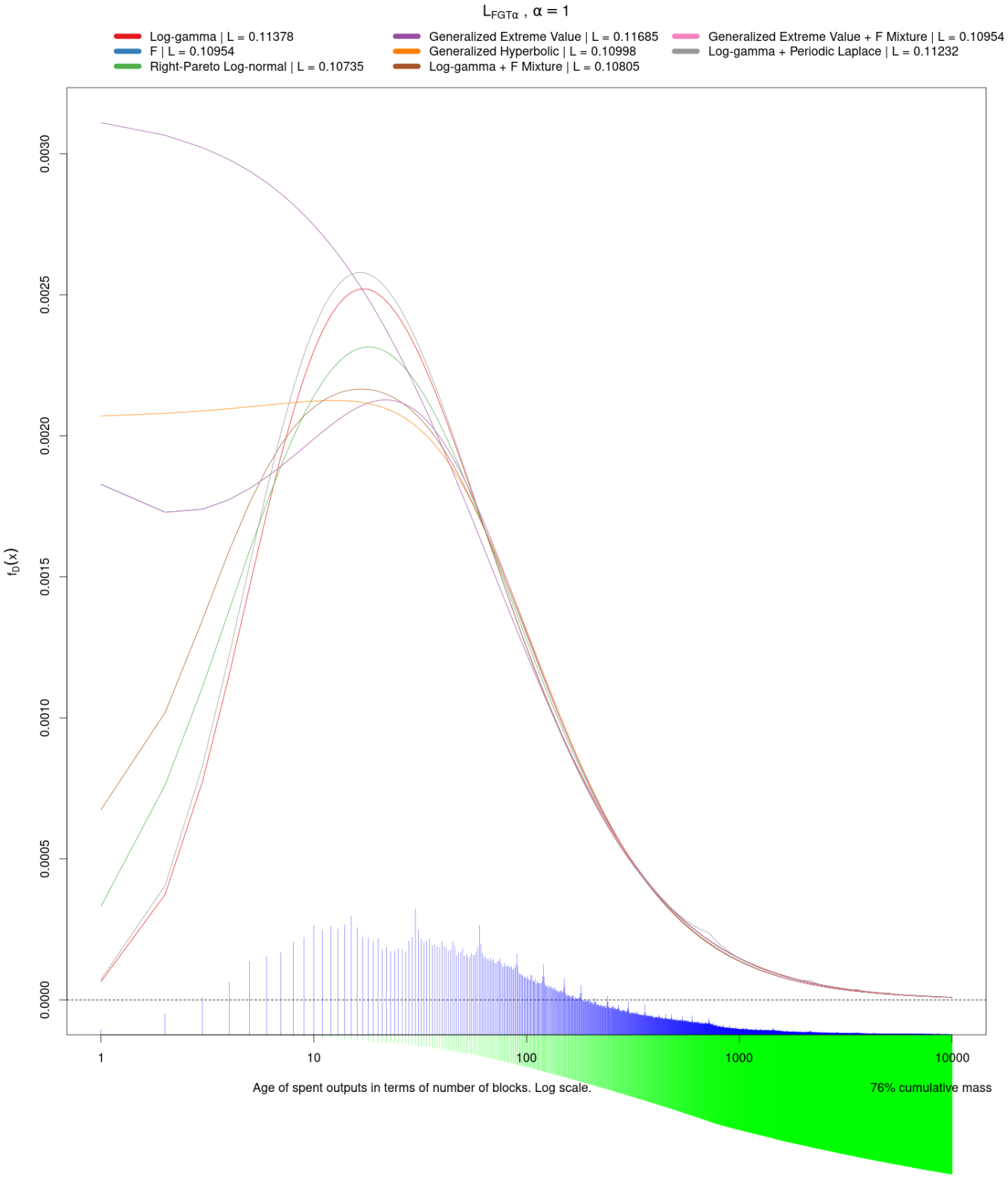


Figure 4: Optimized $f_D(x)$ for loss function L_{FGT_α}, $\alpha = 2$

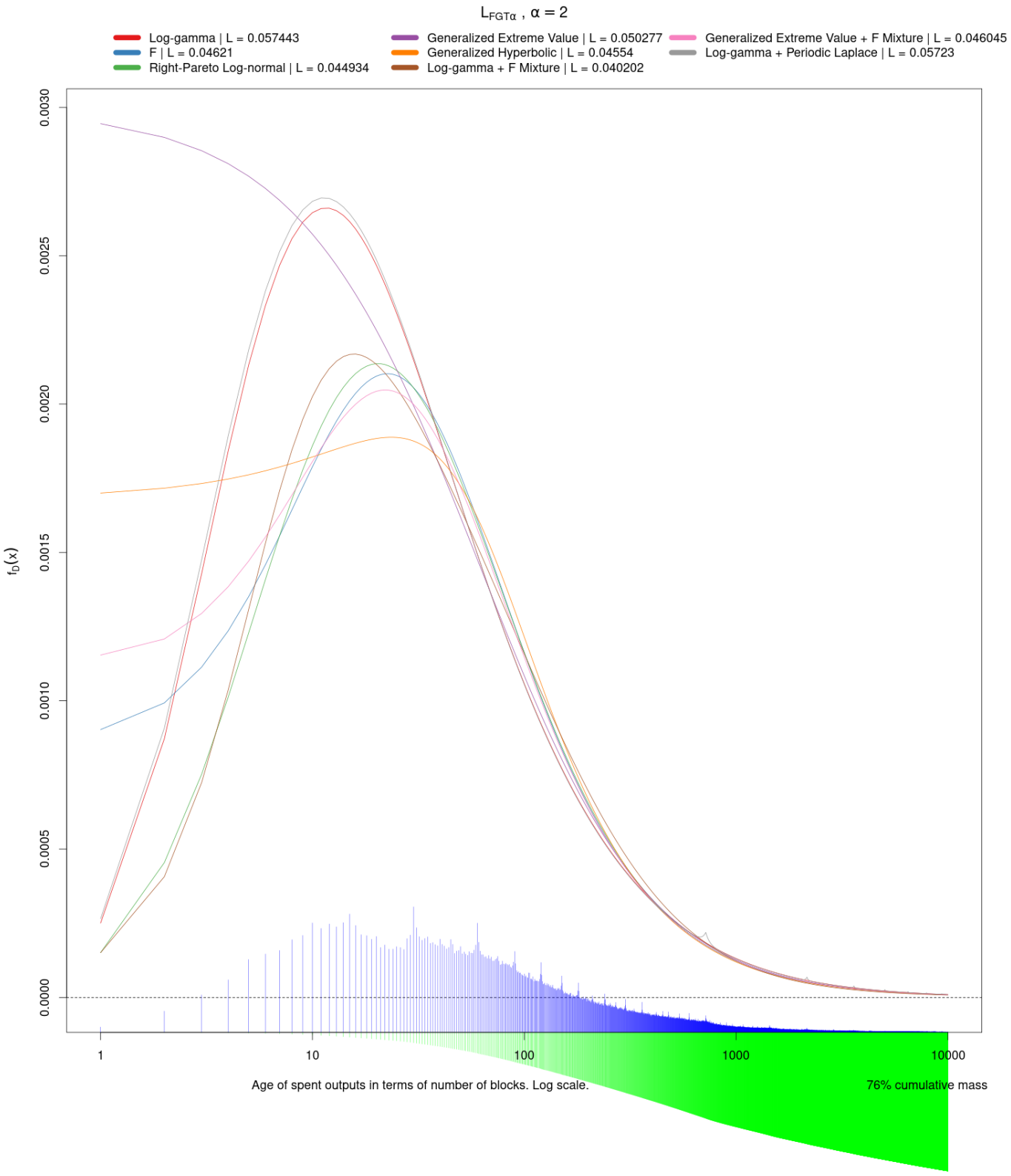


Figure 5: Optimized $f_D(x)$ for loss function $L_{Welfare_\eta}$, $\eta = 0.5$

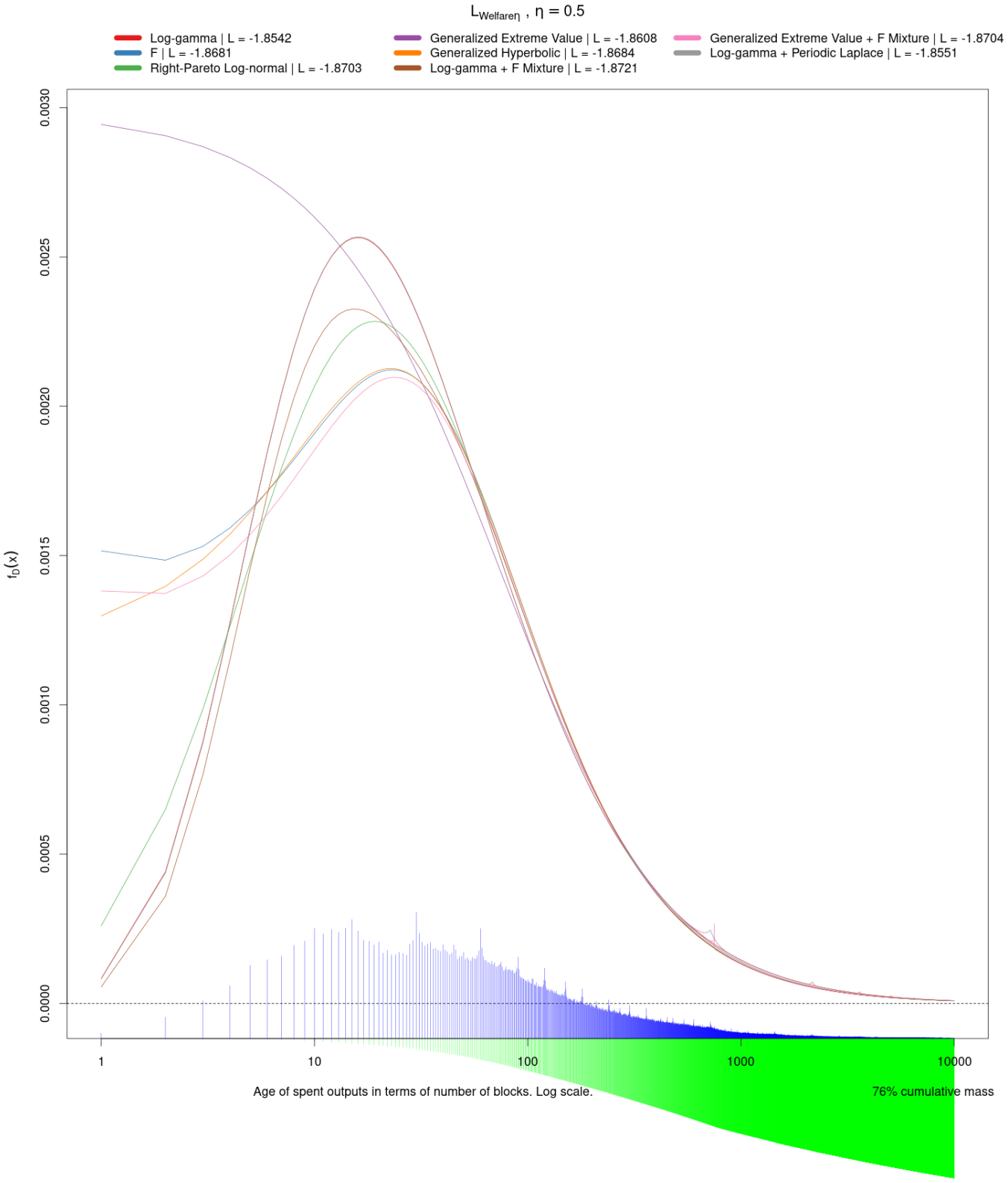


Figure 6: Optimized $f_D(x)$ for loss function $L_{Welfare_\eta}$, $\eta = 1$

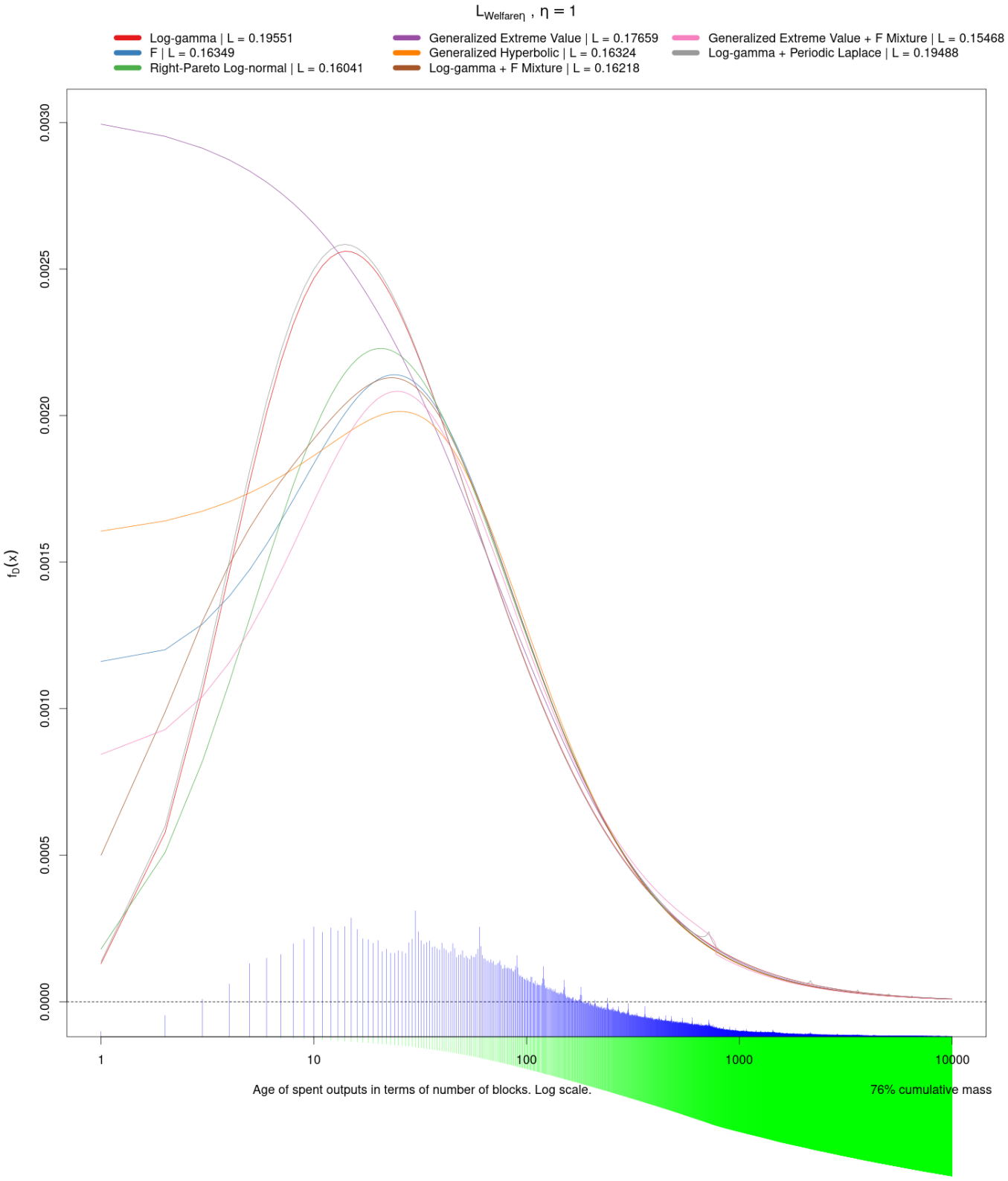


Figure 7: Optimized $f_D(x)$ for loss function L_{MLE}

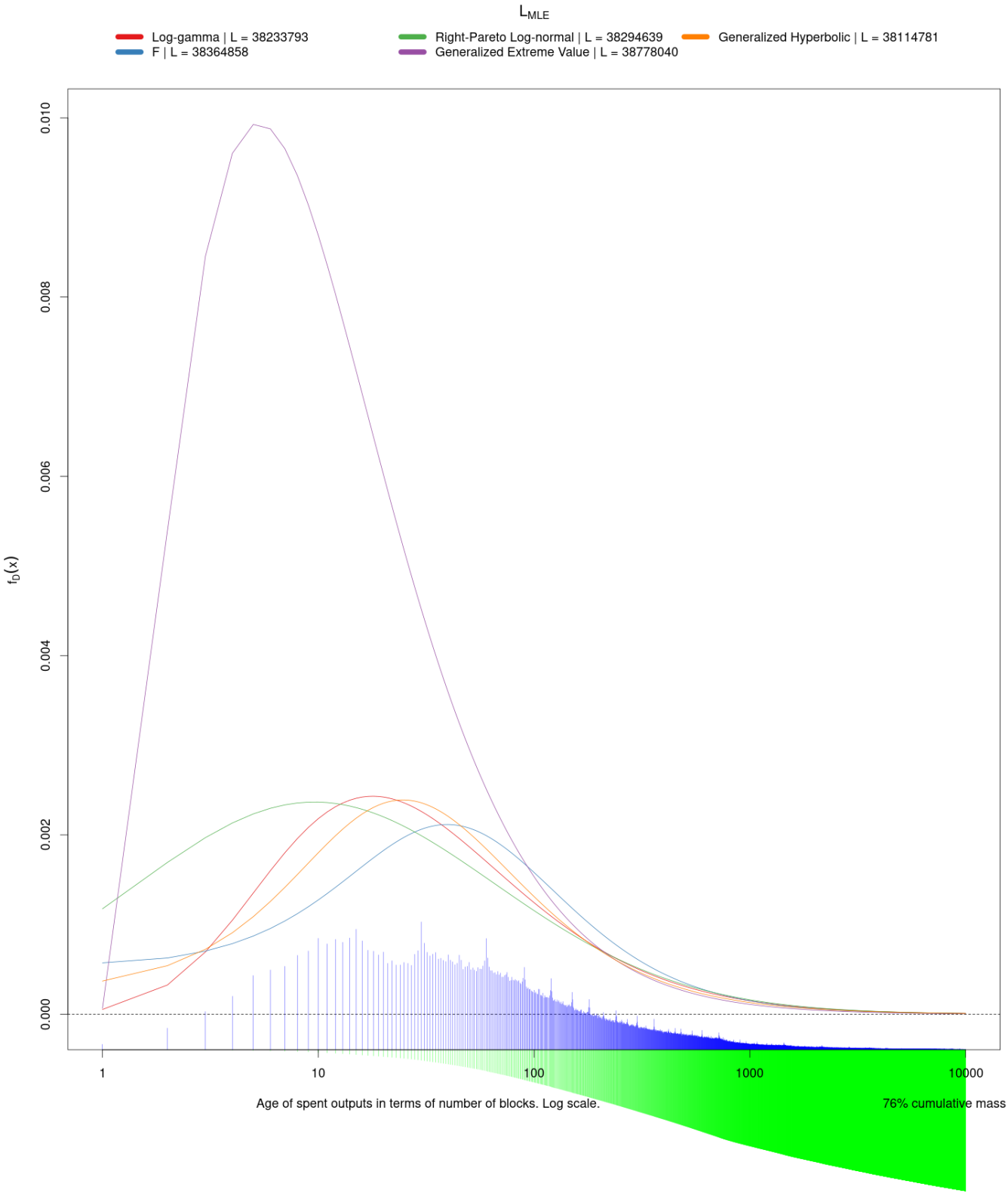


Figure 8: Optimized $f_D(x)/f_S(x)$ for loss function L_{FGT_α}, $\alpha = 1$

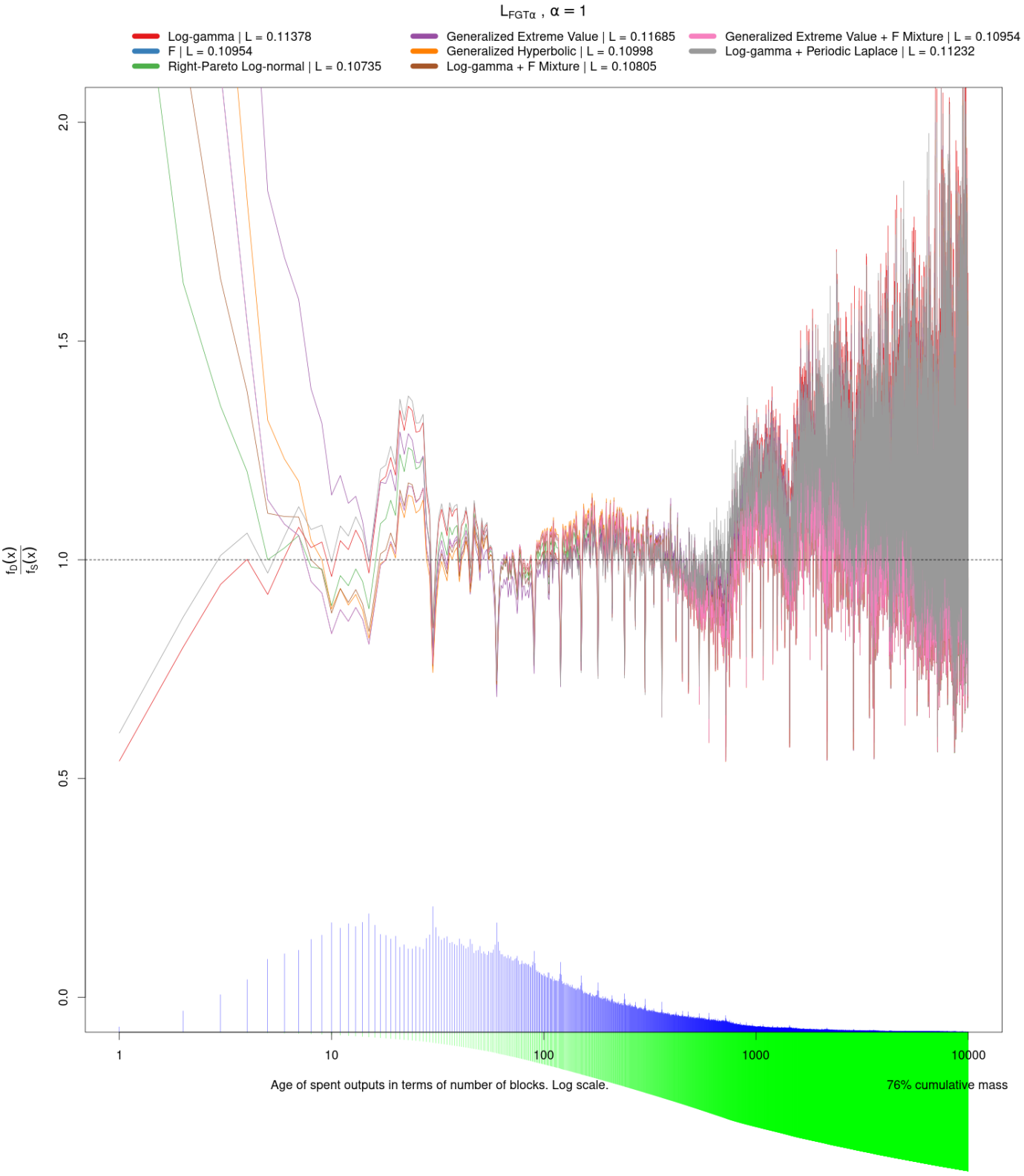


Figure 9: Optimized $f_D(x)/f_S(x)$ for loss function L_{FGT_α}, $\alpha = 2$

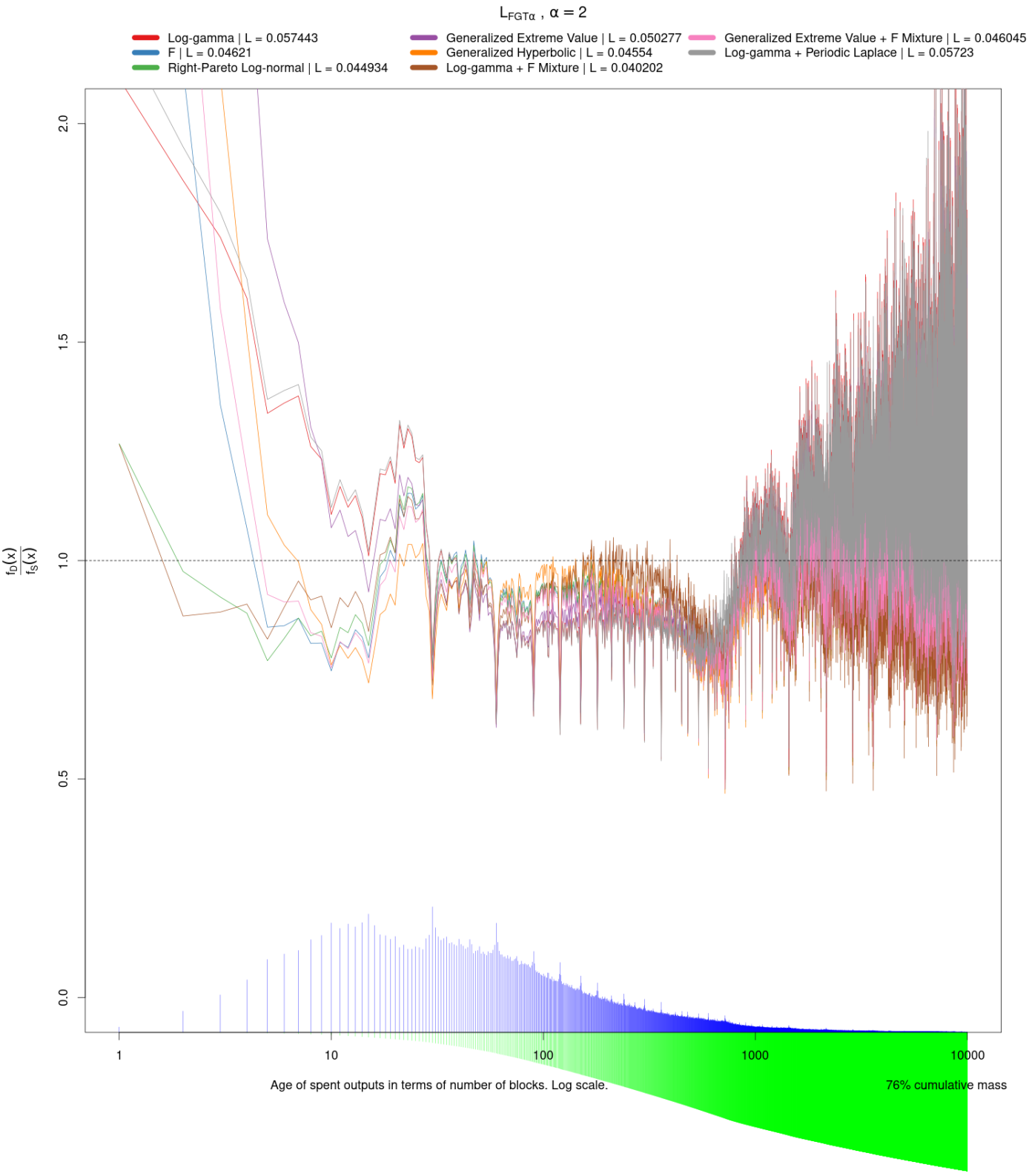


Figure 10: Optimized $f_D(x)/f_S(x)$ for loss function $L_{Welfare_\eta}$, $\eta = 0.5$

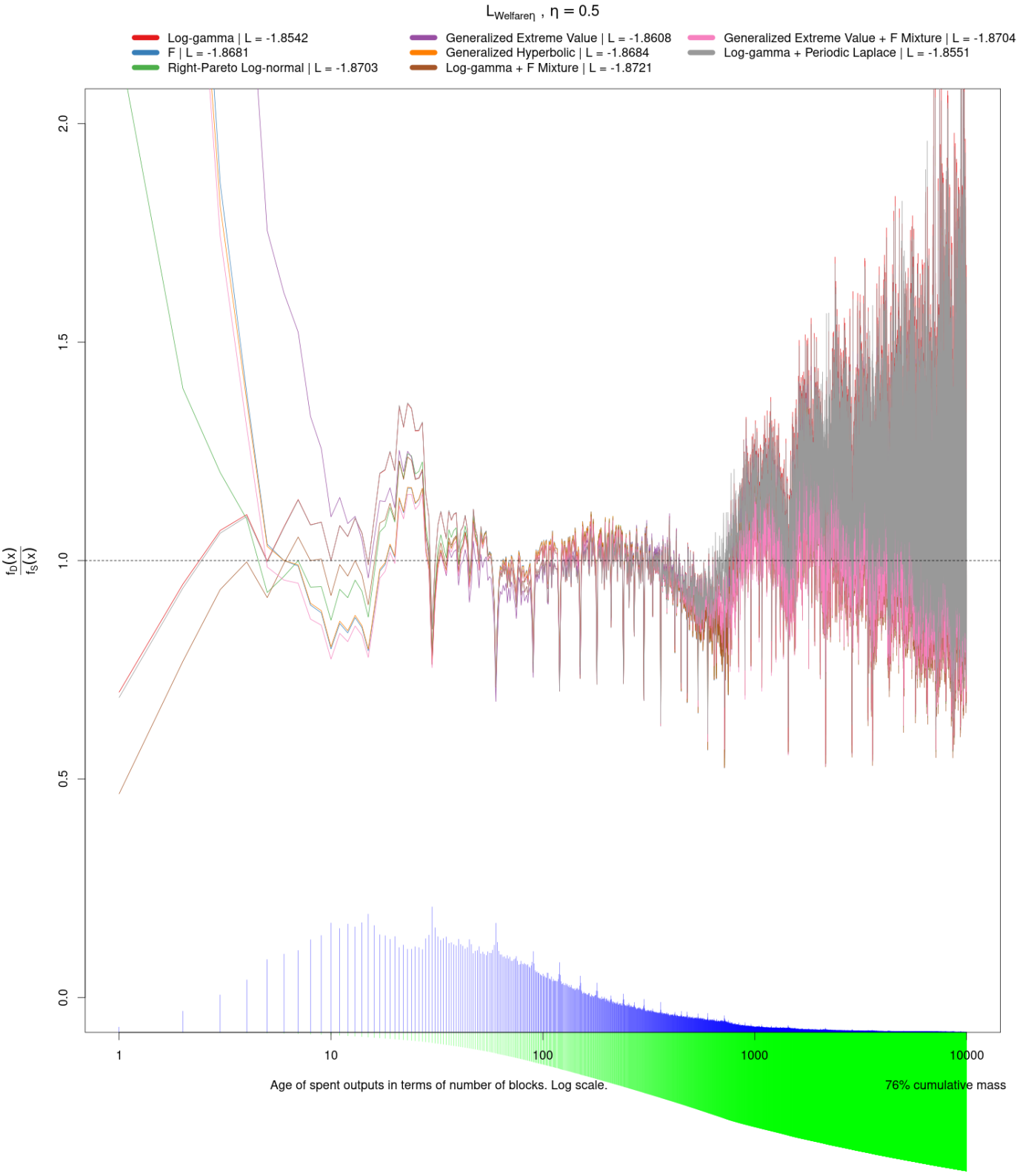


Figure 11: Optimized $f_D(x)/f_S(x)$ for loss function $L_{Welfare_\eta}$, $\eta = 1$

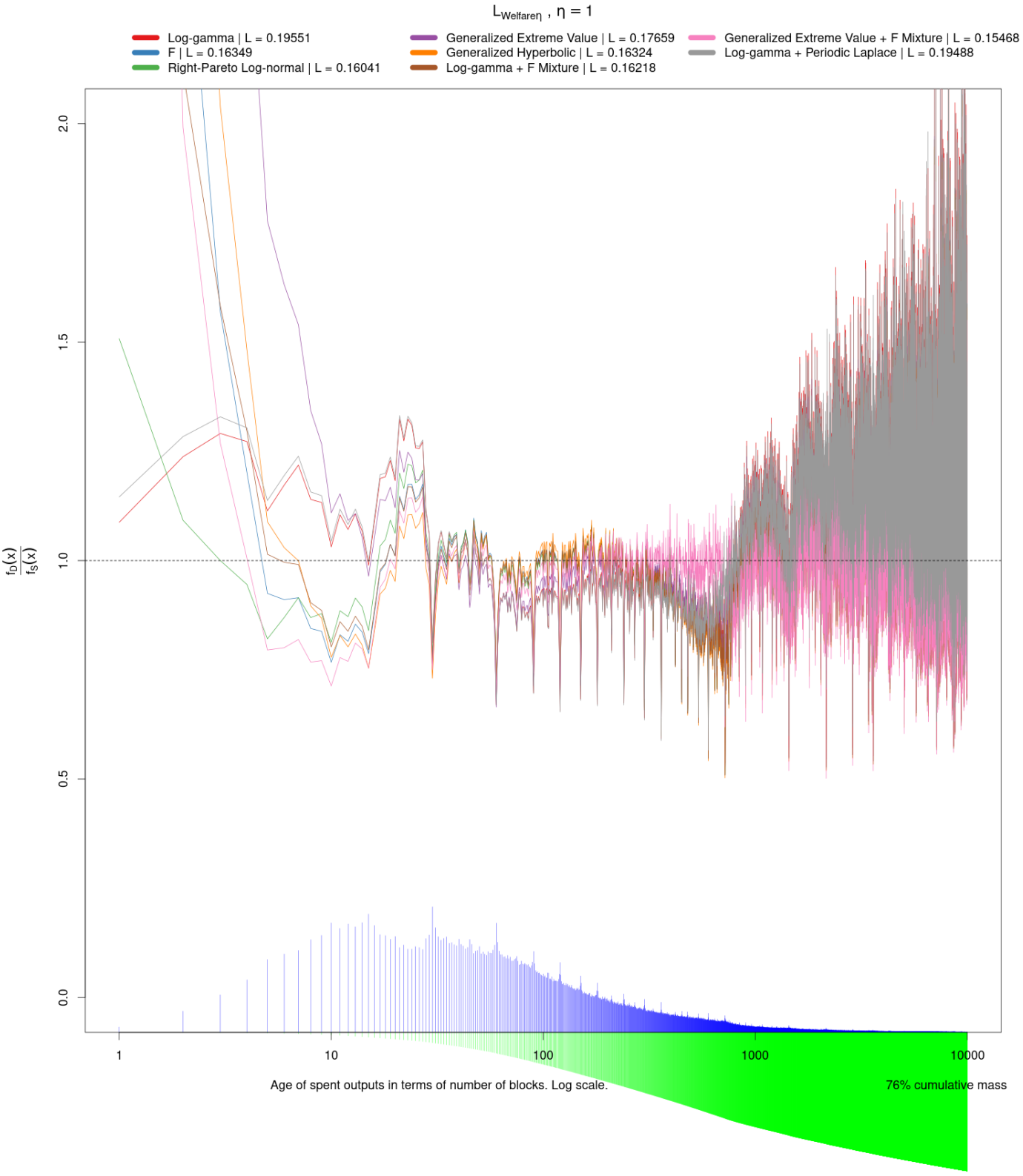
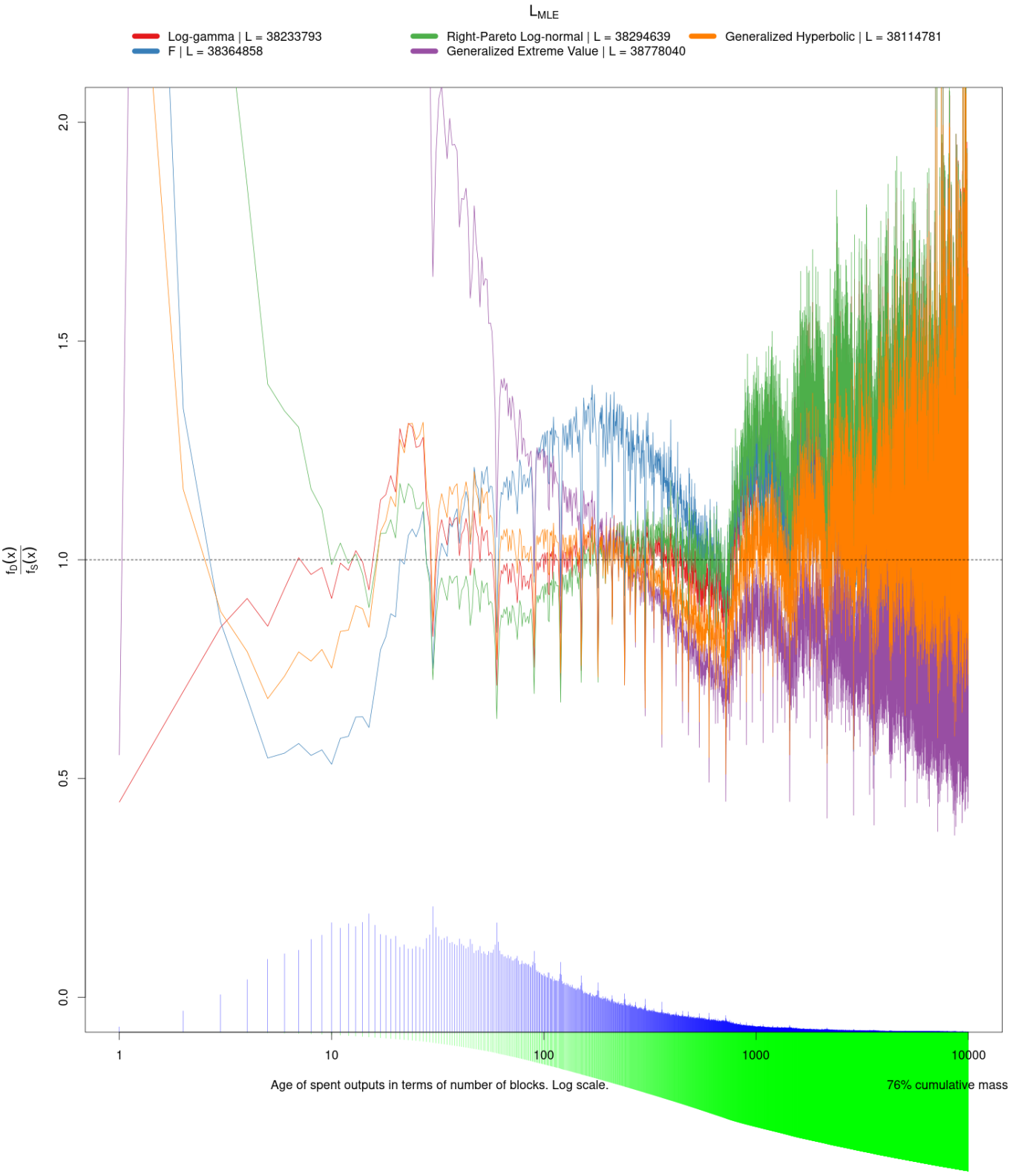


Figure 12: Optimized $f_D(x)/f_S(x)$ for loss function L_{MLE}



6 Options for Loss Functions $\mathcal{L}$ and Parametric Distributions $\mathcal{F}$

In **Section 4 Criteria for Best Fit** I developed several loss functions that could be used to decide which parametric distribution (and particular parameter values) should be used to form the OSPEAD decoy selection algorithm. At this point in time, it is not necessary to decide which loss function to use. We can perform the minimizations, examine the results, and make a decision when we have all the information.

1. Privacy impoverishment. This loss function overall appears to be one of the better choices. It is intuitive and provides options in its α parameter. To minimize risk of very low coverage of certain intervals on the probability distribution, setting $\alpha \geq 2$ would probably be best.

2. Economic welfare. This criteria is less intuitive than privacy impoverishment, but it offers another way to adjust the risk sensitivity with its η parameter. Setting $\eta \geq 1$ will force more risk avoidance.

3. Inequality minimization. This criteria would be most important if we seek to minimize inequality among users. One of its main drawbacks is that is can be computationally expensive.

4. Worst-case-scenario minimization. This criteria has some appeal because worst case scenarios seem to be a focus in the field of cryptography. However, focusing on the worst case out of hundreds of thousands may be questionable.

5. Maximum Likelihood Estimation. [Möser et al., 2018] used this criteria. For reasons stated above, I do not recommend this one.

6. Maximize resistance to a specific attack. This criteria has some intuitive appeal: defend against an attack directly. However, it is difficult to say whether maximizing resistance to one attack may leave users vulnerable to a different attack.

There is no particular statistical theoretical reason to choose one parametric distribution over another. Issues with software implementation may push us to use a simpler and more common distribution. If wallet developers do not use the `wallet2` implementation, they may have difficulty implementing a distribution defined by a mixture of unusual distributions.

7 Dynamic Risk and Forecasting

The Monero Research Lab research bulletin that I quoted in the introduction continues with the following observation [Mackenzie et al., 2015]:

> This would suggest that we, the developers of Monero, must estimate the probability distribution governing the age of transaction outputs.
>
> This, too, is problematic. When an exchange rate is experiencing a strong long-term decline (inflation), rational users are more likely to spend their transaction outputs, for tomorrow their outputs will be worth less in terms of goods and services than today and hoarding is not economically rational. When an exchange rate is experiencing a strong long-term increase (deflation), rational users are more likely to hoard their transaction outputs for the opposite reason. Hence, the distribution of transaction output ages will at least vary over time, and, presuming any proportion of users are rational will certainly depend sensitively on the economic performance of the currency. It is unwise to design security recommendations around the economic performance of our protocol.

I would argue that, as long as Monero is committed to using a mimicking decoy selection algorithm, "design[ing] security recommendations around the economic performance of our protocol" is unavoidable, if unfortunate. The best that we can do is to minimize the risk of guessing the future distribution incorrectly. This requires an assessment of dynamic risk with forecasting. According to empirical evidence from transparent chains, the spent output age distribution is affected more by volatility than long-term inflation and deflation. [Makarov & Schoar, 2021] concluded that "the vast majority of Bitcoin transactions between real entities are for trading and speculative purposes. Starting from 2015, 75% of real bitcoin volume has been linked to exchanges or exchange-like entities such as on-line wallets, OTC desks, and large institutional traders." I discussed my own findings for DOGE in the June 1, 2022 Monero Research Lab meeting.[6] The transaction patterns of Monero may be somewhat different due to its absence from many centralized exchanges, but probably not completely different. Assume for a moment that changes in the real spend age distribution are entirely caused by exchange rate volatility. In that case, forecasting the real spend age distribution would be roughly equivalent to forecasting exchange rate volatility and therefore would be highly challenging. The Efficient Market Hypothesis would imply that a naive forecast may be the best forecast.

We return to Equation (2) from **Section 4 Criteria for Best Fit**: the risk function $R(\theta, \delta) = \mathrm{E}_\theta L(\theta, \delta(\mathbf{X}))$. This is the expected value of our choice of the loss function when a particular set of parameters, e.g. the shape, scale, and location of a parametric distribution, are set to particular values. The theoretical bounds on $R(\theta, \delta)$ can be very wide in our setting and do not grant us much guidance. Therefore, it is best to compute some sort of empirical risk function based on data. All we have is past data, yet our goal is to compute future risk. We must forecast future data and then determine performance of our various options of decoy selection algorithms through forecast validation.

How to do forecasting and forecast validation? A naive, simple method to forecast future values is to assume that the values will be the same as in the current period. More sophisticated methods attempt to anticipate changes in the forecasted values by analyzing cycles and statistical dependence of the values. In our setting with multivariate forecasting, some appropriate forecasting methods include Exponential Smoothing, Vector Autoregression (VAR), Vector Autoregressive Moving Average (VARMA), Generalized Autoregressive Conditional Heteroskedasticity (GARCH), and Kalman Filter.

Cross-validation is a popular general method to evaluate model accuracy. There are special considerations when the data is time series because time series data is not necessarily independent nor identically distributed. The general consensus in the academic literature is that when the characteristics of the time series are unknown then "out-of-sample" (OOS) cross-validation should be used. OOS cross-validation fits a forecasting model on

[6] https://libera.monerologs.net/monero-research-lab/20220601#c103366

all the data except the last few periods and then measures how well the model forecasts the data of the last few periods. OOS cross-validation is also known as "last block validation", "forward validation", and "holdout validation" ([Bergmeir & Benítez, 2012], [Cerqueira et al., 2020]).

To be specific, what I mean by "unknown" characteristics of a time series is that it is unknown whether the time series is stationary. Stationarity is a technical condition in time series that requires that its distribution does not depend on time. An independent and identically distributed series is stationary. A series that is neither independent nor identically distributed will not be stationary. Many financial time series are non-stationary. A common way to manage non-stationarity in data is to compute the first difference (or second difference if necessary, or third...), but it is not clear how this could be done when the object of observation is an entire distribution, as it is in our setting.

[Bergmeir et al., 2018] mathematically prove that standard K-fold cross-validation is a valid technique when the data series is stationary (plus a few other assumptions). They warn that K-fold cross-validation is not valid if the data series is non-stationary and show through Monte Carlo simulations that OOS validation performs better on non-stationary data. [Bergmeir & Benítez, 2012] arrive at a similar conclusion. [Cerqueira et al., 2020] separate dozens of real data sets into stationary and non-stationary and finds that "when the time series are non-stationary, the most accurate estimates are produced by out-of-sample methods, particularly the holdout approach repeated in multiple testing periods....When the observations in a data set are not i.i.d. [independently and identically distributed], the standard cross-validation approach is not directly applicable." The top performing models in the M5 forecasting competition generally used the last 28 day period for cross-validation ([Makridakis et al., 2022]).

There is no particular reason to think that the spent output age distribution is stationary. Certainly, research into spent output age is in its infancy. Non-stationarity is the weaker assumption. To be "safe" it is better to impose fewer assumptions on statistical models if we cannot justify them. Furthermore, with only about 52 weeks of Monero data it would be difficult to perform a formal hypothesis test of stationarity with high statistical power. Therefore, it is best to use OOS cross-validation to evaluate forecast accuracy and risk. A form of OOS cross-validation is performed in the following section.

The `tsqsim` software created by Monero developer mj-xmr is capable of OOS cross-validation.[7] Some modifications will be necessary to support multivariate time series. Technically, spent output age is univariate, but since we are measuring the age *distribution* at each time period, it makes sense to model it in a multivariate way.

8 Inter-Temporal Stability of Spent Output Age Distribution for BTC, BCH, LTC, and DOGE

Much like in **Section 5 Dry Run with Old Möser et al. (2018) Data**, it is useful to run an analysis of similar spent output age data from other blockchains to inform statistical modeling strategies.

The table below lists the share of payments made with several different cryptocurrencies with an unspent transaction output (UTXO) model. I ignored cryptocurrencies with an account model like Ethereum because the spending mechanism is fundamentally different. The main question that we want to answer is whether the chosen cryptocurrencies are used in a roughly similar manner to Monero: as peer-to-peer electronic cash. This table may suggest that BTC, BCH, LTC, and DOGE have somewhat similar usage as Monero. One major difference between these cryptocurrencies and Monero is that there is a 10 block lock enforced at the protocol level for Monero. BTC, BCH, LTC, and DOGE allow users to spend received funds immediately.

[7] https://github.com/mj-xmr/tsqsim

UTXO-based cryptocurrency usage as payment by percent (payment processors and merchants)							
	Service	Bitpay[8]	CoinCards[9]	Bitrefill[10]	Cake Pay[11]	Travala[12]	Shodan[13]
	TXs/month	67,000	NA	186,000	NA	NA	220 total
	Time period	June 2022	July 2022	NA	Aug 2022	NA	NA
	Metric	Transactions	USD value	Unique users	USD value	Hotel nights	Subscriptions
BTC		53.3	41.0	40	16	9.2	47.3
BCH		5.0	NA	NA	NA	0.5	4.1
LTC		21.2	7.0	7	6	0.8	12.7
DOGE		6.2	3.0	NA	NA	0.3	13.2
XMR		NA	19.0	NA	78	1.9	NA
DASH		NA	0.5	NA	NA	0.5	NA
ADA		NA	NA	NA	NA	0.9	NA

In the next several section I perform statistical analysis of these four UTXO cryptocurrencies. The code is available at https://github.com/Rucknium/OSPEAD/tree/main/General-Blockchain-Age-of-Spent-Outputs. The code was executed on the Monero Research Computing Server, whose hardware was improved by a CCS proposal.[14]

8.1 Summary Characterization of Distributions Over Time

Figures 14 to 17 show several statistics about the age of spent outputs of BTC, BCH, LTC, and DOGE since 2015. The age units are in terms of blocks. For BTC and BCH the interval between blocks is 10 minutes. For LTC it is 2.5 minutes and for DOGE it is 1 minute. The unit of observation is the ISO week, a natural unit of economic time.[15]

The first line graphs show the mean, median, standard deviation, skewness, and kurtosis. The skewness and kurtosis statistics may be unfamiliar. They are the third and fourth standardized moments of a distribution, respectively. The moments of a distribution is defined by a power of the expectation $\mathrm{E}[X]$, i.e. theoretical mean, of the distribution. The kth moment is $\mathrm{E}[X^k]$. Moments extend the concept of moving from expectation to variance (which is the square of the standard deviation). The mean (expectation) of a random variable is simply the first moment:

$$\mu = \mathrm{E}[X^1]$$

The variance is the second central moment:

$$\sigma^2 = \mathrm{E}\left[(X - \mathrm{E}[X])^2\right]$$

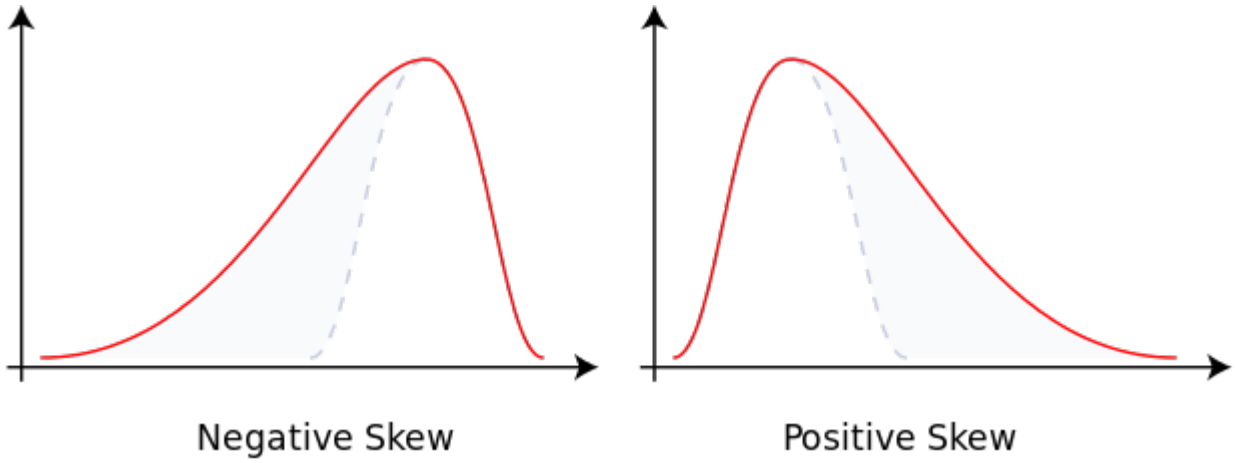
The skewness is the third standardized moment (standardized by the standard deviation σ):

$$\tilde{\mu}_3 = \mathrm{E}\left[\left(\frac{X - \mathrm{E}[X]}{\sigma}\right)^3\right]$$

The kurtosis is the fourth standardized moment:

[8] https://bitpay.com/stats/
[9] https://twitter.com/CoinCards/status/1555286172385116164
[10] https://youtu.be/bkjEcSmZKfc?t=549
[11] https://twitter.com/cakewallet/status/1565370179906838528
[12] https://travala-dashboard.com/
[13] https://blog.shodan.io/accepting-crypto-a-vendor-perspective/
[14] https://ccs.getmonero.org/proposals/gingeropolous_zenith_storage.html
[15] https://en.wikipedia.org/wiki/ISO_week_date

Figure 13: Skewness



Source: https://en.wikipedia.org/wiki/File:Negative_and_positive_skew_diagrams_(English).svg

$$\tilde{\mu}_4 = \mathrm{E}\left[\left(\frac{X - \mathrm{E}[X]}{\sigma}\right)^4\right]$$

The mean is a measure of the central tendency of a distribution. The standard deviation is a measure of its dispersion (spread). The skewness and kurtosis of a distribution involve its characteristics in its tail or tails. A positive skew means that the distribution's tail is on the right side of the distribution. All the age distributions analyzed here tend to have a positive skew. High kurtosis means that large outliers are more likely. Kurtosis of greater than 3 suggests that a distribution has a fatter tail than the normal distribution. The skewness and kurtosis become relevant when very old outputs "wake up" during periods of exchange rate volatility to participate in speculative activity, i.e. buying and selling on exchanges.

The fitting function is essentially a minimum divergence estimator. This means that the parameters of the parametric probability density function (PDF) are chosen to minimize the distance between the parametric PDF and the PDF formed by the data (the empirical PDF), for some specified metric of "distance".

There are several measures of distance that could be used. For the purpose of this exploration of output age distribution forecasting, the distance metric to be minimized will be the total linear sum of the mass of the estimated parametric PDF that falls below the empirical PDF. With the "loss function" specified this way, the optimization algorithm attempts to minimize the probability that the real spends are much more likely to come from a block of a particular age compared to a potential decoy. Ideally, the decoy distribution would be identical to the real spend age distribution, but parametric PDFs are not flexible enough to perfectly match an empirical PDF.

Define $f_S(x)$ as the empirical spent output age distribution at block age x and $f_D(x;\boldsymbol{\beta})$ as a potential "decoy" distribution with some parameter vector $\boldsymbol{\beta}$ (with the transparent blockchains presented here, there is no actual decoy mechanism of course). Then for each week of spent output age data the parameter vector $\boldsymbol{\beta}$ can be chosen to minimize this quantity:

$$L(\boldsymbol{\beta}) = \sum_{i \in \{x_i : f_D(x_i;\boldsymbol{\beta}) < f_S(x_i)\}}^{N} f_S(x_i) - f_D(x_i;\boldsymbol{\beta})$$

That is, minimize the sum of the difference between the real spend age distribution for blocks x_i where the decoy distribution is less than the real spend age distribution. The minimization is performed by computer numerical minimization methods similar to gradient descent.

The two candidate "decoy" parametric PDFs under consideration in this exercise are the Log-gamma (*lgamma*)

distribution and the Right-Pareto Log-normal (*rpln*) distribution. The lgamma distribution, with two parameters, was used in [Möser et al., 2018] to suggest a decoy distribution that was later incorporated into Monero's reference wallet software. The rpln distribution is a more flexible distribution, with three parameters. The PDFs of these two distributions are:

$$f_{lgamma}(x) = \frac{ratelog^{shapelog}}{\Gamma(shapelog)} \times \frac{(\ln x)^{shapelog-1}}{x^{ratelog+1}}$$

with $ratelog > 0$ and $shapelog > 0$ and where Γ is the gamma function and

$$f_{rpln}(x) = shape2 \times x^{-shape2-1} e^{shape2 \times meanlog + \frac{shape2^2 \times sdlog^2}{2}} \Phi\left(\frac{\ln x - meanlog - shape2 \times sdlog^2}{sdlog}\right)$$

with $shape2 > 0$ and $sdlog > 0$ and where Φ is the cumulative distribution function of the standard Normal distribution.

Figure 14: BTC Summary Characterization of Spend Age Distribution Over Time

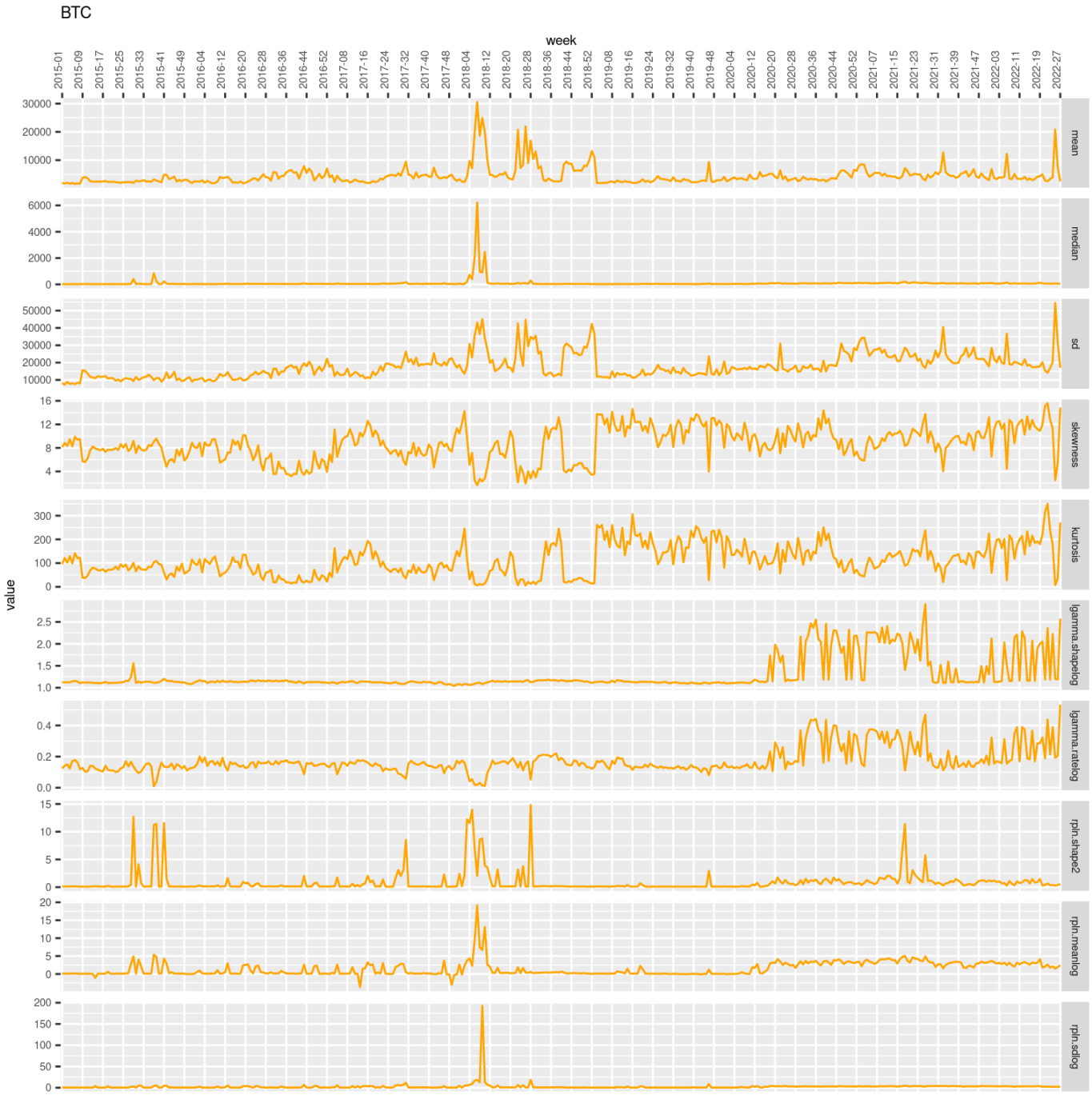


Figure 15: BCH Summary Characterization of Spend Age Distribution Over Time

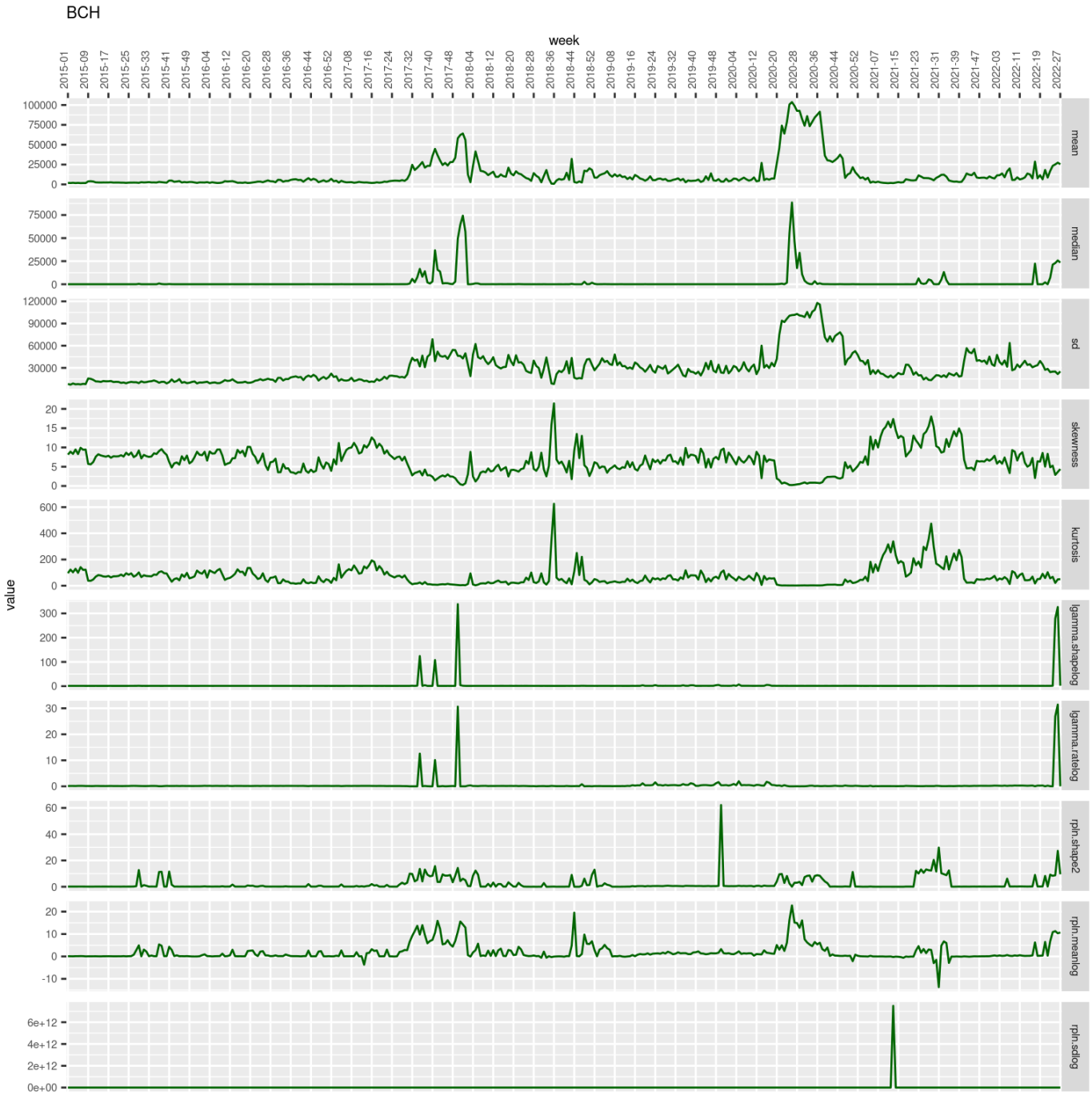


Figure 16: LTC Summary Characterization of Spend Age Distribution Over Time

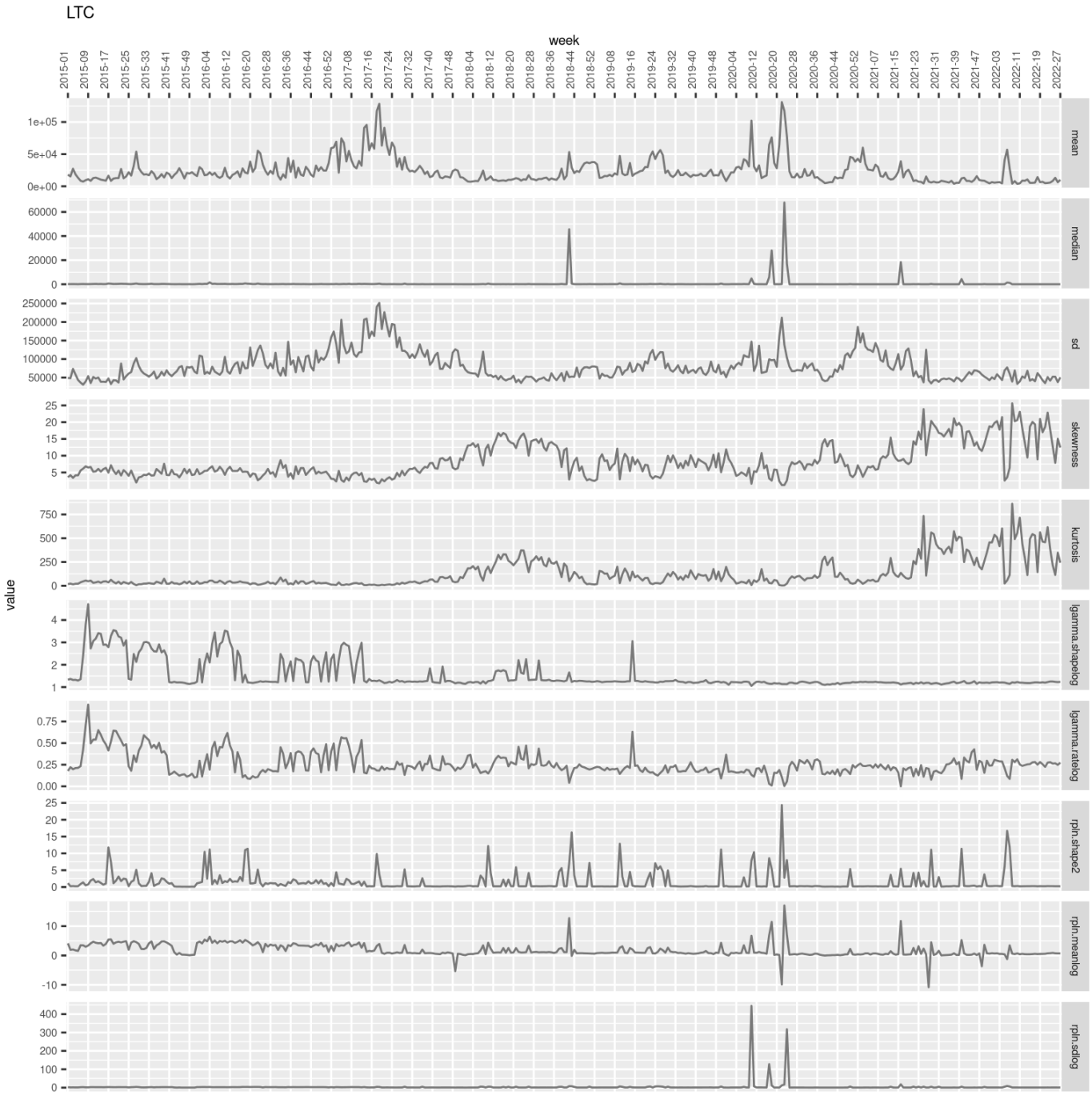
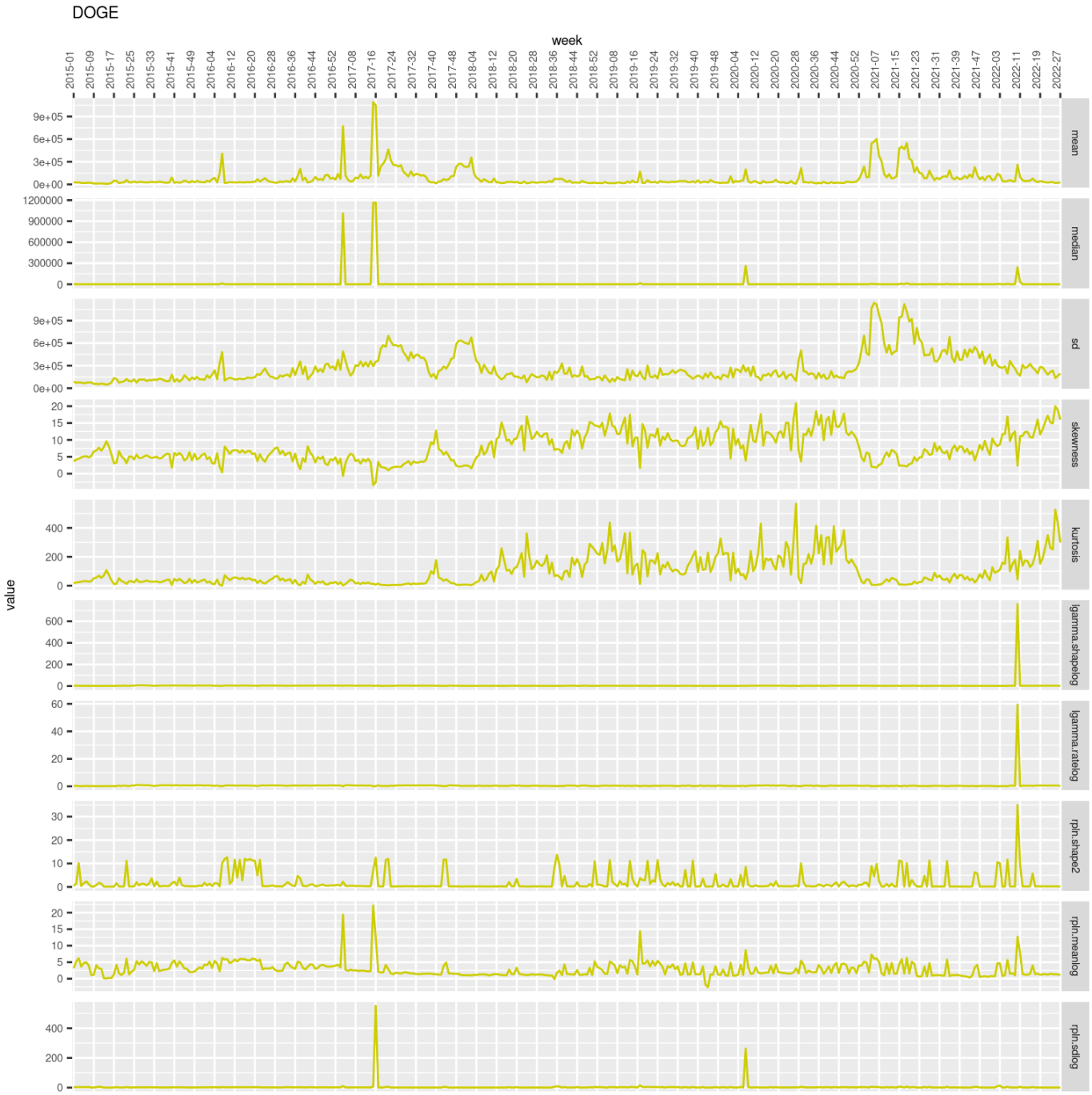


Figure 17: DOGE Summary Characterization of Spend Age Distribution Over Time



8.2 Cross-Blockchain Correlations of Summary Statistics Across Time

The following table contains the correlation between several statistics over time for each pair of blockchains. The "BTC&BCH" correlation included data only for January 2018 onward to avoid artificially raising the correlation by including weeks when the BTC and BCH contained the same transaction before the August 2017 hard fork.

To be precise, define vector $[x_{s,1}, x_{s,2}, \ldots, x_{s,Z}] = \mathbf{X}_s$ for statistic s, e.g. the median, of blockchain x at each week, with Z total weeks in the sample. Define $\mathbf{Y}_s$ for blockchain y similarly. Then the quantities displayed in Table 4 are $corr(\mathbf{X}_s, \mathbf{Y}_s)$.

The two statistics that tend to have consistently high correlation for each pair of blockchains are skewness and kurtosis.

Table 4: Cross-Blockchain Correlations of Summary Statistics Across Time

	BTC&BCH	BTC&LTC	BTC&DOGE	BCH&LTC	BCH&DOGE	LTC&DOGE
mean	-0.04	-0.04	-0.04	-0.01	-0.08	0.21
median	-0.03	-0.01	-0.01	-0.03	-0.02	-0.01
sd	-0.07	0.02	0.34	-0.02	0.02	0.35
skewness	0.09	0.24	0.19	0.11	-0.32	0.28
kurtosis	0.02	0.29	0.31	0.22	-0.26	0.23

8.3 Evaluation of Forecast Accuracy

OSPEAD needs to approximate the real spend age distribution in the future, not the past. Ideally, the decoy selection algorithm should mimic the real spend age distribution at the time that each transaction is made. Since the real spend age distribution is likely changing over time, some type of forecasting is needed. Here I perform some simple evaluation of forecasting methods on transparent blockchains.

Let W be the number of weeks in a forecast horizon. It is the approximate interval of time that a proposed decoy selection algorithm should aim to be accurate. Let M be the number of weeks in the data sample. Then a common way to evaluate a forecasting method is to use the first $M - W$ weeks to fit a forecasting model. Then determine the performance of the method by forecasting for W weeks into the "future" and compare with the actual sample data of the final W weeks. This is known as out-of-sample validation. For this exercise, I set $W = 8$.

An entire distribution needs to be forecast rather than a single value or a set of values. Forecasting an entire (empirical, nonparametric) distribution is extremely complex, so here I reduce the magnitude of the forecasting problem to just the parameters of the fitted parametric distributions (lgamma and rpln).

As an initial test, the sophisticated forecasting method used here was a multivariate auto-regressive(1) exogenous inputs state-space model with a Koopman-Durbin Kalman filter implemented in the `MARSS` R package. The forecasting method allows a forecast into each period in the future. Since the "S" in OSPEAD stands for "Static", only one of the forecast periods can be chosen. I chose the 4th forecast period since it is roughly in the middle of the 8-week forecast horizon. This forecast is referred to as simply `forecast.accuracy` below.

I also evaluated two naive forecasting methods. The first is to use the parameters of the $M - W$th week, i.e. the last week of the "training set", as the forecast. This is called `forecast.accuracy.naive.final.week` below. The second naive method (`forecast.accuracy.naive.horizon.period`) is to use the final W weeks of the training set to fit the specified lgamma and rpln distributions by minimizing the $L(\boldsymbol{\beta})$ loss function for a set of weeks of data pooled together rather than a single week.

Given that the loss function $L(\boldsymbol{\beta})$ is not globally convex in the choice parameters $\boldsymbol{\beta}$, the numerical optimization algorithm can go "off the rails" and settle at a local rather than global minimum. Such a failure mode is especially likely if the empirical distribution is unusual or if the starting values given to the optimization algorithm are far from the global minimum. Generally, the solution to this problem is to try different optimization algorithms, determined better starting values, or use a computationally expensive grid search method. I chose to defer dealing with the issue to a later stage in the process. For the purposes of forecasting, I removed any weeks where the estimated parameters of the parametric distributions were greater than (or less than) five times the 95th (5th) percentile from the median. For example, this exclusion step caused the forecast evaluation for BCH to use the 15th through 24th weeks of 2022 rather than the 20th through 27th week as for other blockchains.

The results of the forecast evaluation are in Table 5. Forecasts were evaluated based on the value of their loss function $L(\boldsymbol{\beta})$ for each out-of-sample week. Lower values indicate better performance. The minimum possible value

is 0 and the maximum possible value is 1. The color code scales were calculated separately for each blockchain. Each color claims a value bin of equal size. Higher values are red, average values are yellow, and low values are green.

In this preliminary exercise, `rpln.forecast.accuracy.naive.horizon.period` appears to have the most consistently good performance across blockchains.

Table 5: Evaluation of forecast accuracy

BTC

rownames	mean	sd	2022-20	2022-21	2022-22	2022-23	2022-24	2022-25	2022-26	2022-27
rpln.forecast.accuracy	0.1225	0.0294	0.0952	0.1086	0.1058	0.1227	0.0926	0.1814	0.1319	0.1421
rpln.forecast.accuracy.naive.final.week	0.1213	0.0295	0.0951	0.1068	0.1037	0.1222	0.0910	0.1807	0.1313	0.1397
rpln.forecast.accuracy.naive.horizon.period	0.0682	0.0244	0.0795	0.0504	0.0541	0.0601	0.0501	0.1237	0.0645	0.0632
lgamma.forecast.accuracy	0.1256	0.0215	0.1216	0.1103	0.1151	0.1149	0.1059	0.1733	0.1299	0.1341
lgamma.forecast.accuracy.naive.final.week	0.1495	0.0277	0.1179	0.1373	0.1373	0.1519	0.1188	0.1996	0.1613	0.1717
lgamma.forecast.accuracy.naive.horizon.period	0.1338	0.0177	0.1302	0.1183	0.1274	0.1212	0.1205	0.1716	0.1358	0.1453

BCH

rownames	mean	sd	2022-15	2022-16	2022-18	2022-19	2022-20	2022-21	2022-22	2022-24
rpln.forecast.accuracy	0.2416	0.1923	0.1811	0.1420	0.1147	0.1598	0.1057	0.4079	0.1665	0.6547
rpln.forecast.accuracy.naive.final.week	0.2452	0.1808	0.1882	0.1530	0.1302	0.1668	0.1208	0.3957	0.1692	0.6375
rpln.forecast.accuracy.naive.horizon.period	0.3938	0.0463	0.3837	0.3886	0.3754	0.3830	0.3809	0.3812	0.3526	0.5051
lgamma.forecast.accuracy	0.3654	0.1374	0.2606	0.2757	0.2574	0.3479	0.2506	0.3830	0.5578	0.5906
lgamma.forecast.accuracy.naive.final.week	0.3573	0.1491	0.2401	0.2553	0.2367	0.3476	0.2313	0.3867	0.5643	0.5967
lgamma.forecast.accuracy.naive.horizon.period	0.4179	0.0769	0.3675	0.3757	0.3720	0.3858	0.3473	0.4209	0.5200	0.5539

LTC

rownames	mean	sd	2022-20	2022-21	2022-22	2022-23	2022-24	2022-25	2022-26	2022-27
rpln.forecast.accuracy	0.0950	0.0182	0.0736	0.0849	0.0718	0.1054	0.1139	0.1184	0.0859	0.1060
rpln.forecast.accuracy.naive.final.week	0.0947	0.0183	0.0748	0.0864	0.0691	0.1036	0.1123	0.1197	0.0859	0.1059
rpln.forecast.accuracy.naive.horizon.period	0.0883	0.0150	0.0762	0.0827	0.0682	0.0951	0.1015	0.1108	0.0745	0.0974
lgamma.forecast.accuracy	0.1112	0.0181	0.0873	0.1014	0.0899	0.1231	0.1328	0.1317	0.1035	0.1201
lgamma.forecast.accuracy.naive.final.week	0.1120	0.0181	0.0880	0.1025	0.0906	0.1239	0.1336	0.1324	0.1044	0.1208
lgamma.forecast.accuracy.naive.horizon.period	0.1056	0.0162	0.0853	0.0949	0.0893	0.1168	0.1247	0.1248	0.0949	0.1144

DOGE

rownames	mean	sd	2022-20	2022-21	2022-22	2022-23	2022-24	2022-25	2022-26	2022-27
rpln.forecast.accuracy	0.2410	0.0202	0.2446	0.2479	0.2635	0.2692	0.2431	0.2137	0.2300	0.2161
rpln.forecast.accuracy.naive.final.week	0.1928	0.0203	0.2044	0.2070	0.2104	0.2179	0.1903	0.1626	0.1799	0.1697
rpln.forecast.accuracy.naive.horizon.period	0.1888	0.0199	0.2089	0.2049	0.1995	0.2100	0.1786	0.1548	0.1773	0.1761
lgamma.forecast.accuracy	0.2093	0.0177	0.2329	0.2231	0.2186	0.2154	0.1935	0.1770	0.2037	0.2106
lgamma.forecast.accuracy.naive.final.week	0.2050	0.0174	0.2139	0.2153	0.2205	0.2262	0.2012	0.1728	0.1967	0.1934
lgamma.forecast.accuracy.naive.horizon.period	0.1953	0.0177	0.2127	0.2091	0.2040	0.2125	0.1859	0.1619	0.1904	0.1856

8.4 Animated Evolution of Distributions

I created gif animations of the empirical probability densities of each blockchain's spent output age distribution over time. The green line is the current week's empirical density, with faint echoes of prior weeks in other colors. The fitted lgamma distribution line is white and rpln is red. The horizontal axis of the first charts for each blockchain has a log scale. Both the horizontal and vertical charts are log scale for the second chart of each blockchain. The "1" on the horizontal axis should be interpreted as "0". In other words, a "1" means that the age of the spent output is approximately zero. This would occur if (1) a child transaction spent an output from a parent transaction that was confirmed in the same block as the child transaction; (2) the timestamps of blocks were out of order, leading to a "negative" age; (3) the block was confirmed at a time interval less than half of the target block time, e.g. 5 minutes for BTC/BCH, leading to rounding down to zero.

There is a clear daily cycle visible in the log-log scale charts.

To view the animated gifs, visit:

https://rucknium.me/html/spent-output-age-btc-bch-ltc-doge.html

9 Options for Forecasting

My suggestion is to reduce the dimensionality of the forecasting problem by forecasting the parameters of parametric distributions, as in **Section 8.3 Evaluation of Forecast Accuracy**. In this case we would need forecasting methods that can handle multivariate forecasting objectives. Beyond that requirement, there are few theoretical reasons to favor particular forecasting methods in this setting. Out-of-sample forecasting validation should guide the final choice of forecast method. The one exception is forecasting methods that technically require the time series to be stationary. We may want to somewhat discount such methods and choose one only if it offers much superior performance to a method that allows non-stationarity.

Forecasting methods under consideration include:

1. Exponential Smoothing. The `legion` R package can perform multivariate Exponential Smoothing.
2. Vector Autoregression (VAR). The `MTS` R package can perform VAR forecasting through its `VARpred` function. Note that VAR generally requires stationary time series unless explicit cointegration relationships are specified as in a Vector Error Correction model.
3. Vector Autoreregressive Moving Average (VARMA). The `MTS` R package can perform VAR forecasting through its `VARMApred` function. Generally, stationarity is required.
4. Generalized Autoregressive Conditional Heteroskedasticity (GARCH). The `rmgarch` R package offers multivariate GARCH forecasting. Generally, stationarity is required.
5. Kalman Filter. The `MARSS` and `KFAS` R packages provide multivariate Kalman Filter forecasting methods.
6. A naive forecast where the forecast is just the value of the parameters of the last period in the training set.
7. A naive forecast where the parameters are fit on multiple weeks of pooled data on the last part of the training set.

A simple way to evaluate forecast accuracy is to compute the loss function on some final W number of weeks as I did in **Section 8.3 Evaluation of Forecast Accuracy**. Another option is to perform the evaluation on a rolling window, called "evaluation on a rolling forecasting origin" in Section 5.10 "Time series cross-validation" of [Hyndman & Athanasopoulos, 2021]. For now for notational simplicity we will assume the former method.

Let $\mathcal{Q}$ be the set of forecasting methods listed above and $q(\boldsymbol{\beta})$ be a particular element of the $\mathcal{Q}$ set. $q(\boldsymbol{\beta})$ is a function that takes as arguments fitted parameter values $\hat{\boldsymbol{\beta}}$ of parametric PDFs $f(x|\boldsymbol{\beta}) \in \mathcal{F}$ of past weekly values and outputs forecasts $\tilde{\boldsymbol{\beta}}$ for future $\boldsymbol{\beta}$. Considering the fitting procedure from Equation (13), the final decoy selection algorithm parametric distribution $f(x|\boldsymbol{\beta})$ and parameters $\boldsymbol{\beta}$ should be chosen by:

$$\underset{f(x|\boldsymbol{\beta})\in\mathcal{F}, q\left(\hat{\boldsymbol{\beta}}\right)\in\mathcal{Q}}{\arg\min} \frac{1}{W}\sum_{i=1}^{W} L\left(f\left(x_i \middle| q\left(\hat{\boldsymbol{\beta}}\right)\right)\right), \forall q \in \mathcal{Q} \tag{14}$$

Expression (14) will find the minimizers $f(x|\tilde{\boldsymbol{\beta}})$ of each loss function $L \in \mathcal{L}$ among all parametric distributions $f(x|\boldsymbol{\beta}) \in \mathcal{F}$ and all forecasting methods $q(\boldsymbol{\beta}) \in \mathcal{Q}$ of the mean for the out-of-sample data x_i for weeks $i = 1, \ldots, W$. Once these $f(x|\tilde{\boldsymbol{\beta}})$ are found, the only decision left to be made is the judgment call on the proper choice of loss function and any associated loss function parameters λ, α, η. For example, in Table 5, Expression (14) would choose `rpln.forecast.accuracy.naive.horizon.period` for BTC, LTC, and DOGE, and `rpln.forecast.accuracy` for BCH. Note that Expression (14) forms a plug-in estimator for the risk function in Equation (2). Therefore, it is a type of empirical risk minimizer.[16]

10 Acknowledgments

This research was funded by Monero's Community Crowdfunding System:

https://ccs.getmonero.org/proposals/Rucknium-OSPEAD-Fortifying-Monero-Against-Statistical-Attack.html

The following people gave feedback on the research process and/or contributed in other ways: ACK-J, ArticMine, bob, coinstudent2048, garth, gingeropolous, isthmus, jberman, kayabaNerve, koe, mj-xmr, monerobull, moneromooo, neptune, plowsof, SamsungGalaxyPlayer, SerHack, SethForPrivacy, and Syksy.

[16] https://en.wikipedia.org/wiki/Empirical_risk_minimization#Empirical_risk_minimization

References

[Bergmeir & Benítez, 2012] Bergmeir, C. & Benítez, J. M. (2012). On the use of cross-validation for time series predictor evaluation. *Information Sciences*, 191, 192–213. https://doi.org/https://doi.org/10.1016/j.ins.2011.12.028. Data Mining for Software Trustworthiness

[Bergmeir et al., 2018] Bergmeir, C., Hyndman, R. J., & Koo, B. (2018). A note on the validity of cross-validation for evaluating autoregressive time series prediction. *Computational Statistics & Data Analysis*, 120, 70–83. https://doi.org/https://doi.org/10.1016/j.csda.2017.11.003

[Casella & Berger, 2002] Casella, G. & Berger, R. L. (2002). *Statistical Inference* (second ed.). Duxbury Pacific Grove, CA.

[Cerqueira et al., 2020] Cerqueira, V., Torgo, L., & Mozetič, I. (2020). Evaluating time series forecasting models: an empirical study on performance estimation methods. *Machine Learning*, 109(11), 1997–2028. https://doi.org/10.1007/s10994-020-05910-7

[Creedy, 2015] Creedy, J. (2015). A Note on Computing the Gini Inequality Measure with Weighted Data. Working Paper Series 4235, Victoria University of Wellington, Chair in Public Finance. https://ideas.repec.org/p/vuw/vuwcpf/4235.html

[Hsu et al., 2014] Hsu, J., Gaboardi, M., Haeberlen, A., Khanna, S., Narayan, A., Pierce, B. C., & Roth, A. (2014). Differential privacy: An economic method for choosing epsilon. *2014 IEEE 27th Computer Security Foundations Symposium*. https://doi.org/10.1109/csf.2014.35

[Hyndman & Athanasopoulos, 2021] Hyndman, R. J. & Athanasopoulos, G. (2021). *Forecasting: Principles and Practice* (third ed.). OTexts: Melbourne, Australia. https://otexts.com/fpp3/

[Krawiec-Thayer et al., 2021] Krawiec-Thayer, M. P., Neptune, Rucknium, Jberman, & Carrington (2021). *Fingerprinting a flood: forensic statistical analysis of the mid-2021 monero transaction volume anomaly.* https://mitchellpkt.medium.com/fingerprinting-a-flood-forensic-statistical-analysis-of-the-mid-2021-monero-transaction-volume-a19cbf41 Available at https://mitchellpkt.medium.com/fingerprinting-a-flood-forensic-statistical-analysis-of-the-mid-2021-monero-transaction-volume-a19cbf41ce60

[Kumar et al., 2017] Kumar, A., Fischer, C., Tople, S., & Saxena, P. (2017). A traceability analysis of monero?s blockchain. *European Symposium on Research in Computer Security (ESORICS)*. https://doi.org/10.1007/978-3-319-66399-9_9

[Mackenzie et al., 2015] Mackenzie, A., Noether, S., & Team, M. C. (2015). *Improving obfuscation in the cryptonote protocol*. Research Bulletin. https://www.getmonero.org/resources/research-lab/pubs/MRL-0004.pdf

[Maji et al., 2019] Maji, A., Ghosh, A., Basu, A., & Pardo, L. (2019). Robust statistical inference based on the c-divergence family. *Annals of the Institute of Statistical Mathematics*, 71(5), 1289–1322. https://doi.org/10.1007/s10463-018-0678-5

[Makarov & Schoar, 2021] Makarov, I. & Schoar, A. (2021). Blockchain analysis of the bitcoin market. Working Paper 29396, National Bureau of Economic Research. https://doi.org/10.3386/w29396

[Makridakis et al., 2022] Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2022). M5 accuracy competition: Results, findings, and conclusions. *International Journal of Forecasting*. https://doi.org/https://doi.org/10.1016/j.ijforecast.2021.11.013

[Möser et al., 2018] Möser, M., Soska, K., Heilman, E., Lee, K., Heffan, H., Srivastava, S., Hogan, K., Hennessey, J., Miller, A., Narayanan, A., & Christin, N. (2018). An empirical analysis of traceability in the monero blockchain. *Proceedings on Privacy Enhancing Technologies*, 2018(3), 143–163. https://doi.org/doi:10.1515/popets-2018-0025

[Reed & Jorgensen, 2004] Reed, W. J. & Jorgensen, M. (2004). The double pareto-lognormal distribution—a new parametric model for size distributions. *Communications in Statistics - Theory and Methods*, 33(8), 1733–1753. https://doi.org/10.1081/STA-120037438

[Ronge et al., 2021] Ronge, V., Egger, C., Lai, R. W. F., Schröder, D., & Yin, H. H. F. (2021). Foundations of ring sampling. *Proceedings on Privacy Enhancing Technologies*, 2021(3), 265–288. https://doi.org/doi:10.2478/popets-2021-0047

[Vijayakumaran, 2021] Vijayakumaran, S. (2021). *Analysis of cryptonote transaction graphs using the dulmage-mendelsohn decomposition*. Cryptology ePrint Archive, Report 2021/760. https://eprint.iacr.org/2021/760

[Ye et al., 2020] Ye, C., Ojukwu, C., Hsu, A., & Hu, R. (2020). *Alt-coin traceability*. Cryptology ePrint Archive, Report 2020/593. https://eprint.iacr.org/2020/593