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

Rucknium

September 21, 2022

Delivered to the OSPEAD Scientific Review Panel for Milestone 1 of the OSPEAD CCS proposal

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

2 Overview of the plan

OSPEAD is a two-part statistical technique. The two parts can each be separated into two steps. The first part produces an estimate of the historical real spend age distribution. The first part is potentially useful to an adversary who wishes to harm Monero users' privacy, so a consideration of disclosure is necessary before releasing it publicly. The second part takes the real spend age distribution as an input and produces a usable parametric distribution for Monero's decoy selection algorithm, taking into account variability in the real spend age distribution over time. The second part can be released publicly. Therefore, I intend to detach the second part of this document and release it publicly so the community can examine progress of OSPEAD and give feedback.

The two steps of the first part are:

1. Estimate the sub-distributions formed by the diverse decoy selection algorithms that various wallet software use.

2. Use the sub-distribution associated with the standard `wallet2` decoy selection algorithm to estimate the real spend age distribution.

The two steps of the second part are:

1. Fit a parametric distribution to the real spend age distribution estimated in part 1 by using a loss function minimization approach.

2. Use the time variability of the historical real spend age distribution and forecast error risk as feedback into a final parametric decoy selection algorithm.

The estimation procedure I propose in this document follows the outline in my HackerOne submission closely, with two main exceptions:

In the submission I noted "the fact that $f_S(x)$ [the real spend age distribution] is literally a moving target" since it evolves over time, but I did not propose any solution to this problem. In this document I propose some forecast validation methods.

In my HackerOne submission I suggested that a Maximum Likelihood Estimator could be developed to handle the multiple decoy selection algorithms (DSAs), modeling them as a mixture distribution. In this document I take a different path. There is still the notion of DSAs forming a mixture distribution, but with my proposed estimator we do not need to know the form of all the DSAs being used in the wild, as would have been required for my original idea. My new proposed approach explicitly leverages the hierarchical grouped ring structure of the data,

which provides much greater ability to identify and estimate all the necessary statistical objects. I am not sure if my original model is even identifiable.[1]

I only considered methods within the frequentist statistical paradigm rather than the Bayesian paradigm since I am not very familiar with Bayesian methods.

3 The Meaning of OSPEAD

OSPEAD is an acronym for Optimal Static Parametric Estimation of Arbitrary Distributions. "Estimation of Arbitrary Distributions" is the main term. "Static" and "Parametric" are qualifiers for the "Optimal" descriptor. The qualifiers are used in the same way that Ordinary Least Squares (OLS) is a Best Linear Unbiased Estimator (BLUE) of the conditional mean of a random variable, as stated by the Gauss-Markov Theorem. BLUE means that among estimators that are both linear and unbiased, OLS is the "best", i.e. has the lowest variance, of any of them. I will work backwards through the terms:

3.1 "Estimation of Arbitrary Distributions"

OSPEAD is an estimator of arbitrary distributions. In this case, the distributions are the weekly real spend age distributions of transaction outputs on the Monero blockchain. I say "arbitrary" here since I do not impose strict assumptions on the form of the distributions.

3.2 "Parametric"

OSPEAD is parametric in the sense that the final decoy selection algorithm will be defined by a parametric rather than nonparametric probability density function. The current decoy selection algorithm is formed by a parametric probability density function. It may be the case that a nonparametric probability density function would be better than a parametric one, but the advisability and feasibility of a nonparametric one is left for future research. This is consistent with a Monero development principle: "The Monero Core Team and the Monero Research Lab would like to follow the development philosophy that it is wise to start with smaller changes at first and then ramp those changes up over time, rather than start with drastic changes and try to scale them back." [Mackenzie et al., 2015] Even though the final form of the OSPEAD decoy selection algorithm will be parametric, there are intermediate steps that will involve nonparametric estimation.

3.3 "Static"

OSPEAD is static rather than dynamic. OSPEAD will recommend that wallet software draws ring members in the same way regardless of any shifts in the blockchain data over time, as it does currently. This is consistent with the cautious development philosophy of Monero. A dynamic decoy selection algorithm could re-estimate the "current" real spend age distribution at any point in time on-the-fly. The advisability and feasible of a dynamic decoy selection algorithm would require further research. However, OSPEAD will take into account the historical variability of the real spend age distribution over time and will attempt to produce good performance despite the fact that the real spend age distribution will change in the future.

[1] In the $f(x) = (1-\alpha)f_S(x) + \sum_{i=1}^{N} \alpha_i f_{D,i}(x)$ model with $\sum_{i=1}^{N} \alpha_i = \alpha$ and each $f_{D,i}(x)$ known, it could be true that the mixing proportions α_i are identifiable by the argument of [Hall, 1981] and the unknown nonparametric $f_S(x)$ might be identifiable by the argument of [Patra & Sen, 2016]. Then the whole model might be identifiable. After finding the literature on repeated measurements in finite mixture models, I decided not to pursue this model further.

3.4 "Optimal"

I use the term "optimal" somewhat loosely. I do not claim that my proposed estimator of the real spend age distribution has the lowest variance of any possible estimator. I argue that my proposed estimator is consistent (i.e. approaches the true value as the number of rings in the sample rises) and has acceptable variance. See **Section 14 Future Work to Improve the Two-Step Estimator** for a discussion about how my proposed estimator could be improved. The parametric fitting part is optimal in the sense that I search over a large space of possible parametric distributions and their parameters values to minimize (optimize) certain specified loss functions.

If we have a near-optimal estimate and parametric fit of the real spend age distribution, do we have near-optimal protection of Monero user privacy, at least regarding timing-based statistical attacks? [Ronge et al., 2021] argue "yes":

> Furthermore, in Section 5 we show that the definition is robust, in the sense that the anonymity changes only slightly when the signer distribution changes slightly. In particular, if a ring sampler is shown to be good with respect to a close estimate $\hat{\mathcal{S}}$ of the real signer distribution $\mathcal{S}$, then it should also be good with respect to the real signer distribution $\mathcal{S}$....
>
> Later in this work, we will show that for 1-signer distributions $\mathcal{S}$ [i.e. only a single real spend within a ring, which is the case for Monero,] the optimal anonymity is always almost achievable. More concretely, in Section 6.2 we show that there exists a "mimicking" sampler Π_{Mimic} which achieves anonymity $\alpha(\mathsf{S}, \Pi_{\text{Mimic}}) \gtrapprox \frac{1}{2}\lg n$, which is only a constant fraction away from the optimum, assuming minimally that S has at least $\lg n$ bits of min-entropy. Although the mimicking sampler achieves near-optimal anonymity, it is mostly theoretical as it requires the knowledge of the distribution $\mathcal{S}$.

Of course, I disagree with the statement of [Ronge et al., 2021] that this fact is only of theoretical interest. A good estimate of $\mathcal{S}$ *is* feasible.

4 What to Read and What to Skip

This document is lengthy. It is not strictly necessary to read certain sections to arrive at a judgment of what estimation strategy OSPEAD should use if the reader has a very limited time budget. An abridged version of this document would be:

- Section 5 What to Decide
- Section 7 Finite Mixture Models
- Section 8 Modeling the Real Spend and Decoy Distributions as a Mixture
- Section 10 First Step: Bonhomme-Jochmans-Robin Estimator
- Section 11 Second Step: Patra-Sen Inversion Estimator
- Section 12.4 Options for Confidence Interval Estimation
- Section 21 Criteria for Best Fit
- Section 23 Options for Loss Functions $\mathcal{L}$ and Parametric Distributions $\mathcal{F}$
- Section 24 Dynamic Risk and Forecasting
- Section 26 Options for Forecasting

These section can be skipped if desired:

- Section 6 Foundational Statistical Concepts, but I would still recommend reading Section 6.4 Consistency and Section 6.5 Identifiability and Identification if unfamiliar with those concepts
- Section 9 Two-Step Estimation
- Section 12 Confidence Interval Estimation
- Section 13 Sampling Strategy and Data Features that Support Estimation
- Section 14 Future Work to Improve the Two-Step Estimator
- Section 15 Rucknium Ratio Attack/MAP Decoder/Discriminant Analysis
- Section 16 Postestimation Classification of On-Chain Rings into DSAs
- Section 17 Disclosure Considerations
- Section 18 OSPEAD Requests to Monero C++ Developer(s)
- Section 19 Plain English Explanation of the Problem
- Section 20 The Solution
- Section 22 Dry Run with Old M oser et al. (2018) Data
- Section 25 Inter-Temporal Stability of the Spent Output Age Distribution for BTC, BCH, LTC, and DOGE
- Section 27 Acknowledgments

5 What to Decide

The purpose of this document is to give the OSPEAD scientific review panel an opportunity to give input on the estimation plan. The estimation is designed to give a range of candidates for the new decoy selection algorithm, so it is expected that multiple estimation options can be done at this point in time. The OSPEAD CCS has two further milestones: "Deliver initial probability density function to scientific review panel" and "Deliver final version of probability density function to Monero developers." So we can do the estimations, see what we get, and then decide on the winning candidate. I have distilled the choices into these sections:

- Section 8 Modeling the Real Spend and Decoy Distributions as a Mixture, regarding whether Equation (7) should be used as the OSPEAD decoy selection algorithm.
- Section 10.5 Options for the First Step Estimator
- Section 11.1 Options for the Second Step Estimator
- Section 12.4 Options for Confidence Interval Estimation
- Section 23 Options for Loss Functions $\mathcal{L}$ and Parametric Distributions $\mathcal{F}$
- Section 26 Options for Forecasting

It would be good to get early feedback on whether the Bonhomme-Jochmans-Robin Estimator in **Section 10.5 Options for the First Step Estimator** should be used so that work can begin on writing a C++ implementation of it (point 2 of **Section 18 OSPEAD Requests to Monero C++ Developer(s)**).

6 Foundational Statistical Concepts

Here I will review some statistical concepts that will be needed to compare different estimators and describe the assumptions required for their validity.

6.1 Cumulative Distribution Function, Probability Density Function, Probability Mass Function, and Their Support

The following are some definitions of statistical objects, from [Casella & Berger, 2002]:

> **Definition 1.5.1** The *cumulative distribution function* or *cdf* of a random variable X, denoted $F_X(x)$, is defined by

$$F_X(x) = P_X(X \leq x), \quad \text{for all } x.$$

$P_X(X \leq x)$ is the probability that a single value of the random variable X is below x. A CDF monotonically rises from 0 to 1. All random variables have a CDF regardless of whether they are continuous or discrete. Therefore, CDFs are often used when statistical theory attempts to be more general. Another definition from [Casella & Berger, 2002]:

> **Definition 1.5.7** A random variable X is *continuous* if $F_X(x)$ is a continuous function of x. A random variable X is *discrete* if $F_X(x)$ is a step function of x.

The actual concept of time is continuous. Monero transactions are measured in discrete time because they are confirmed on the block chain every two minutes on average. Therefore their exact time distribution is governed by a probability mass function:

> **Definition 1.6.1** The *probability mass function* (*pmf*) of a discrete random variable X is given by

$$f_X(x) = P(X = x) \quad \text{for all } x.$$

The value of a PMF at any point x is the probability that the random variable X will have value x.

> **Definition 1.6.3** The *probability density function* or *pdf*, $f_X(x)$, of a continuous random variable X is the function that satisfies

$$F_X(x) = \int_{-\infty}^{x} f_x(t)\, dt \quad \text{for all } x.$$

In essence the PDF is the first derivative of the CDF. Even though ring member ages are discrete due to the discrete nature of blocks, continuous PDFs can be useful in approximating distributions for creating decoy selection algorithms. [Casella & Berger, 2002] continues:

> [Let] $\mathcal{X} = \{x : f_X(x) > 0\}$. The pdf of random variable X is positive only on the set $\mathcal{X}$ and is 0 elsewhere. Such a set is called the *support set* of a distribution or, more informally, the *support* of a distribution. This terminology can also apply to a pmf or, in general, to any nonnegative function.

Support is an important factor to consider when attempting to distinguish different distributions and different random variables from each other. Intuitively, if the support of two variables overlaps either partially or completely, then it will be more difficult to distinguish real data being generated by them.

6.2 Estimator, Estimate, and Estimand

OSPEAD is an estimator. [Casella & Berger, 2002] give a very general definition of a point estimator, which is the most common type of estimator. First their definition of "statistic":

> **Definition 5.2.1** Let $X_1, \ldots, X_n$ be a random sample of size n from a population and let $T(x_1, \ldots, x_n)$ be a real-valued or vector-valued function whose domain includes the sample space of $(X_1, \ldots, X_n)$. Then the random variable or random vector $Y = T(X_1, \ldots, X_n)$ is called a *statistic*. The probability distribution of a statistic Y is called the *sampling distribution of* Y.
>
> [....]
>
> **Definition 7.1.1** A *point estimator* is any function $W(X_1, \ldots, X_n)$ of a sample; that is, any statistic is a point estimator.
>
> Notice that the definition makes no mention of any correspondence between the estimator and the parameter it is to estimate. While it might be argued that such a statement should be included in the definition, such a statement would restrict the available set of estimators.. Also, there is no mention in the definition of the range of the statistic $W(X_1, \ldots, X_n)$. While, in principle, the range of the statistic should coincide with that of the parameter, we will see that this is not always the case.
>
> There is one distinction that must be made clear, the difference between an estimate and an estimator. An *estimator* is a function of the sample, while an *estimate* is the realized value of an estimator (that is, a number) that is obtained when a sample is actually taken. Notationally, when a sample is taken, an estimator is a function of the random variables $X_1, \ldots, X_n$, while an estimate is a function of the realized values $x_1, \ldots, x_n$.

Besides point estimators, there are interval estimators (e.g. estimates of confidence intervals), estimators of probability density functions, and so forth. Generally the purpose of an estimator is to generate a close approximation of the *estimand*, i.e. the object to be estimated, from the available data in the sample.

6.3 Parametric and Nonparametric Models

The distinction between parametric and nonparametric statistical models is crucial for understanding the problem that OSPEAD aims to solve. The Wikipedia entry for "Parametric statistics" adequately explains the distinction:[2]

> **Parametric statistics** is a branch of statistics which assumes that sample data comes from a population that can be adequately modeled by a probability distribution that has a fixed set of parameters. Conversely a **non-parametric model** does not assume an explicit (finite-parametric) mathematical form for the distribution when modeling the data. However, it may make some assumptions about that distribution, such as continuity or symmetry.

A normal distribution, for example, is completely specified once its mean μ and variance σ^2 parameter are set. At every point x, its probability density is then

$$f\left(x|\mu,\sigma^2\right) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

If a statistician *assumed* that a given random variable had a normal distribution, then the statistician would just need to estimate the distribution's mean and variance to completely describe the random variable. Assuming that variables are distributed in a particular parametric way is done for convenience, computational and theoretical tractability, and sample size reasons.

Parametric distributions often can describe an idealized physical or social process. A normal distribution is the limiting distribution of the sum of an infinite number of random variables. An exponential distribution describes the

[2] https://en.wikipedia.org/wiki/Parametric_statistics

time interval between events for memoryless processes, like proof-of-work block discovery. A binomial distribution defines the probability of n successful Bernoulli trials with probability p. And so forth.

The real spend age distribution is fundamentally an economic process determined by human decision making. The point of OSPEAD is not to develop a strong theoretical economic model for Monero users' behavior. I will let the data speak for itself rather than asking a theoretical model to speak for it. There is, however, an implicit theoretical model: A current or potential Monero user goes about their daily life. At some point the neurons in the user's brain form into a configuration that is interpreted as "I think I will spend some Monero now." The user then broadcasts a Monero transaction. A miner includes it in (usually) the next block. At any given point in time for each Monero user on the planet there is some unknown latent probability that their neurons will attain the "spend Monero now" configuration. That probability may be different for each user — and within each user the probability may change over time.

There is no particular reason to think that the real spend age distribution conforms to a specific tidy parametric distribution. We do not really know why users spend when they spend, nor do we know which "types" of users tend to spend more (or less) often at particular times. Therefore, I take it as given that the real spend age distribution is nonparametric. A basic example of a nonparametric estimator of a distribution is a histogram. A kernel density estimator, which will appear later, is basically a more sophisticated histogram.

6.4 Consistency

Consistency is essentially a requirement of an adequate estimator. [Casella & Berger, 2002] explain:

> The property of consistency seems to be quite a fundamental one, requiring that the estimator converges to the "correct" value as the sample size becomes infinite. It is such a fundamental property that the worth of an inconsistent estimator should be questioned (or at least vigorously investigated).
>
> Consistency (as well as all asymptotic properties) concerns a sequence of estimators rather than a single estimator, although it is common to speak of a "consistent estimator." If we observe $X_1, X_{2,} \ldots$ according to a distribution $f(x|\theta)$, we can construct a sequence of estimators $W_n = W_n(X_1, \ldots, X_n)$ merely by performing the same estimation procedure for each sample size n. For example, $\bar{X}_1 = X_1$, $\bar{X}_2 = (X_1 + X_2)/2$, $\bar{X}_3 = (X_1 + X_2 + X_3)/3$, etc. We can now define a consistent sequence.
>
> **Definition 10.1.1** A sequence of estimators $W_n = W_n(X_1, \ldots, X_n)$ is a *consistent sequence of estimators* of the parameter θ if, for every $\epsilon > 0$ and every $\theta \in \Theta$,
>
> $$\lim_{n\to\infty} P_\theta(|W_n - \theta| < \epsilon) = 1. \quad (1)$$
>
> Informally, (1) says that as the sample size becomes infinite (and the sample information becomes better and better), the estimator will be arbitrarily close to the parameter with high probability, an eminently desirable property. Or, turning things around, we can say that the probability that a consistent sequence of estimators misses the true parameter is small. An equivalent statement to (1) is this: For every $\epsilon > 0$ and every $\theta \in \Theta$, a consistent sequence $W_n$will satisfy
>
> $$\lim_{n\to\infty} P_\theta(|W_n - \theta| \geq \epsilon) = 0.$$
>
> Definition 10.1.1 should be compared to Definition 5.5.1, the definition of convergence in probability. Definition 10.1.1 says that a consistent sequence of estimators converges in probability to the parameter θ it is estimating.

To summarize: a consistent estimator will produce an estimate extremely close to the true value(s) of the statistical object as the sample size become extremely large. "How close?" and "For what sample size?" are questions left to the discussion on variance below. An inconsistent estimator will converge to the wrong value or no finite value at all.

6.5 Identifiability and Identification

Identification is critical for statistical estimation. For a given statistical model, identifiability asks the question of whether it is even possible to create an estimator that could recover the underlying parameters of interest. In basic statistical procedures, identifiability often is not explicitly discussed since it is easy to prove, but it always exists in the background. For more advanced statistical procedures such as those that OSPEAD will use, identification must be discussed explicitly. [Allman et al., 2009] state,

> General formulations of the identification problem were made by several authors, and pioneering works may be found in [27, 28]. The study of identifiability proceeds from a hypothetical exact knowledge of the distribution of observed variables and asks whether one may, in principle, recover the parameters. Thus identification problems are not problems of statistical inference in a strict sense. However, since nonidentifiable parameters cannot be consistently estimated, identifiability is a prerequisite of statistical parameter inference.

Here the authors are using "consistently estimated" in its technical sense defined above. In essence, statistical estimation is mostly hopeless without identifiability. They continue:

> In the following, we are interested in models defined by a family $\mathcal{M}(\Theta) = \{\mathbb{P}_\theta, \theta \in \Theta\}$ of probability distributions on some space Ω, with parameter space Θ (not necessarily finite dimensional). The classical definition of identifiability, which we will refer to as *strict identifiability*, requires that for any two different values $\theta \neq \theta'$ in Θ, the corresponding probability distributions $\mathbb{P}_\theta$ and $\mathbb{P}_{\theta'}$ are different.

In other words, different parameter values *must* produce different data if a model is to be identifiable. [Casella & Berger, 2002] give another definition (pp. 523):

> **Definition 11.2.2** A parameter θ for a family of distributions $\{f(x|\theta) : \theta \in \Theta\}$ is *identifiable* if distinct values of θ correspond to distinct pdfs or pmfs. That is, if $\theta \neq \theta'$, then $f(x|\theta)$ is not the same function of x as $f(x|\theta')$.
>
> Identifiability is a property of the model, not of an estimator or estimation procedure. However, if a model is not identifiable, then there is difficulty in doing inference. For example, if $f(x|\theta) = f(x|\theta')$ then observations from both distributions look exactly the same and we would have no way of knowing whether the true value of the parameter was θ or θ'. In particular, both θ and θ' would give the likelihood function the same value.
>
> Realize that problems with identifiability can usually be solved by redefining a model.

"Redefining a model" can mean many things. There is "partial identification", which involves narrowing down the possible values of θ to a set or a function of parameters. We will set aside partial identification. The redefinition of a model that is more relevant to our setting is to impose additional constraints, assumptions, or "structure" on the model, based on aspects of the data that are known with certainty or high probability. Fortunately, our problem — identification of Monero's real spend age distribution — has enough structure for identification.

In another setting that may be familiar to the reader, ordinary least squares (OLS) regression is not identified when there is linear dependence between the columns of the "independent" variables, i.e. the design matrix $\mathbf{X}$. A typical cause of this would be perfect correlation of a pair of variables. In this case, the OLS estimator would give an infinite number of solutions to any particular parameter set $\boldsymbol{\theta}$ when minimizing the sum of squared errors. The $\mathbf{X}$ matrix does not have full rank, so the inversion procedure $(\mathbf{X}'\mathbf{X})^{-1}$ in the $(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y} = \boldsymbol{\theta}$ formula for OLS would be invalid. $\mathbf{X}'\mathbf{X}$ is not invertible in this case. Put differently, the sum of squared errors function in this case would achieve its minimum for an infinite number of values of $\boldsymbol{\theta}$, causing the function surface to have a "valley" of infinite length and fixed depth.

6.6 Variance of an Estimator

The expression for the variance of a random variable X is:

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

where $\mathrm{E}[X]$ is the expectation, i.e. theoretical mean, of X.

An estimator itself has a variance since it is a function of random data from a sample of the population. In a frequentist paradigm — rather than a Bayesian paradigm — which OSPEAD orients itself within, an estimator's *estimand*, i.e. the parameter(s) to be estimated, are considered to be fixed in Nature with no variability in itself. Generally in practical applications an estimator's standard error, i.e. the square root of the variance of the estimator, is used since confidence intervals can be directly constructed from a standard error when an estimator is normally distributed.

For most estimators, only the asymptotic variance is known. [Casella & Berger, 2002] give a definition:

> In calculating an asymptotic variance, we are, perhaps, tempted to proceed as follows. Given an estimator T_n based on a sample of size n, we calculate the finite-sample variance $\mathrm{Var}\, T_n$, and then evaluate $\lim_{n\to\infty} k_n \mathrm{Var}\, T_n$, where k_n is some normalizing constant. (Note that in many cases $\mathrm{Var}\, T_n \to 0$ as $n \to \infty$, so we need a factor k_n to force it to a limit.)
>
> **Definition 10.1.7** For an estimator T_n, if $\lim_{n\to\infty} k_n \mathrm{Var}\, T_n = \tau^2 < \infty$, where $\{k_n\}$ is a sequence of constants, then τ^2 is called the *limiting variance* or *limit of the variances.*
>
> [Some discussion of the drawbacks of using limiting variance.]
>
> **Definition 10.1.9** For an estimator T_n, suppose that $k_n(T_n - \tau(\theta)) \to n(0, \sigma^2)$ in distribution [i.e. when scaled by k_n, the difference between the estimator and the estimand converges to a normal distribution with mean 0 and variance σ^2]. The parameter σ^2 is called the *asymptotic variance* or *variance of the limit distribution* of T_n.

A very large number of estimators are asymptotically normally distributed. Many estimators are essentially a type of sum. The sum of an infinite number of independent random variables converges to a normal distribution due to the Central Limit Theorem. However, it cannot be simply assumed that an estimator will be asymptotically normal especially when it is complex or unusual. In the vast majority of parametric settings, the k_n normalizing constant series is $\sqrt{n}$. In this case, such estimators are said to be "$\sqrt{n}$-consistent". In essence, the "speed" or "rate" of convergence for these estimators is $\sqrt{n}$. When attempting to nonparametrically estimate an entire distribution, the rate of convergence is typically slower than $\sqrt{n}$. This fact guides later recommendations about appropriate sample size.

6.7 Bias of an Estimator

If variance is a measurement of the precision of an estimator, then bias indicates its accuracy. [Casella & Berger, 2002] explain:

> **Definition 7.3.2** The *bias* of a point estimator W of a parameter θ is the difference between the expected value W and θ; that is, $\mathrm{Bias}_\theta W = \mathrm{E}_\theta W - \theta$. an estimator whose bias is identically (in θ) equal to 0 is called *unbiased* and satisfies $\mathrm{E}_\theta W = \theta$ for all θ.

To be clear, many, many consistent estimators have some bias. With a large sample, some bias may not be concerning since bias generally shrinks as sample size increases. In informal discussions, "bias" is sometimes used when "inconsistency" is meant. For example, "omitted variable bias" actually refers to inconsistency; with an infinite sample size, omitted variable bias will cause an estimator to mis-estimate the true parameter.

With nonparametric density estimation such as kernel density estimation, some bias is generally unavoidable unless variance is allowed to balloon such as with an "undersmoothing" theoretical condition. This is just a particular instance of the classic variance-bias tradeoff in statistics.

7 Finite Mixture Models

Finite mixture models are the centerpiece of my estimation strategy. [McLachlan et al., 2019] describe finite mixtures:

> The probability density function, or probability mass function in the discrete case, of a finite mixture distribution of a p-dimensional random vector $\boldsymbol{Y}$, takes the form
>
> $$f(\boldsymbol{y}) = \sum_{i=1}^{g} \pi_i f_i(\boldsymbol{y}), \tag{2}$$
>
> where the mixing proportions π_i are nonnegative and sum to one and where the $f_i(\boldsymbol{y})$ are the component densities. We refer to the $f_i(\boldsymbol{y};\boldsymbol{\theta}_i)$ as densities, since even if the vector $\boldsymbol{Y}$ is discrete, we can still view the $f_i(\boldsymbol{y})$ as densities by the adoption of counting measure. Typically, the component densities are taken to be known up to a vector $\boldsymbol{\theta}_i$ of parameters. In this case, we can write the mixture density as
>
> $$f(\boldsymbol{y};\boldsymbol{\Psi}) = \sum_{i=1}^{g} \pi_i f_i(\boldsymbol{y},\boldsymbol{\theta}_i), \tag{3}$$
>
> where $\boldsymbol{\Psi} = (\boldsymbol{\xi}^\top, \pi_1, \ldots, \pi_{g-1})^\top$ denotes the vector of unknown parameters and where $\boldsymbol{\xi}_i$ consists of the elements of the $\boldsymbol{\theta}_i$ known a priori to be distinct. Here the superscript $\top$ denotes transposition. In many applications, the component densities $f_i(\boldsymbol{y};\boldsymbol{\theta}_i)$ are taken to belong to the same parametric family, for example, the multivariate normal.

Our setting is univariate since the focus is solely on age as a variable. Therefore, "p-dimensional random vector $\boldsymbol{Y}$" would be a one-dimensional random variable Y.

In general, identification of the mixing proportions π_i and component densities $f_i(y)$ is quite difficult without more structure imposed on the model. This should not be surprising. If the $f_i(y)$ components are fully nonparametric and their respective supports overlap fully, there is no clear way to use a set of single empirical draws from $f(y)$ to recover π_i and $f_i(y)$. In existing work, researchers have suggested that estimating the real spend age distribution of Monero may be impossible. [Ronge et al., 2021] remarks, "Due to the anonymous nature of anonymous cryptocurrencies, it is (supposedly) infeasible to learn the real signer distributions." [Ni et al., 2021] states, "adversaries cannot guess the probability distribution of consumed tokens." The conclusion of [Aeeneh et al., 2021] claims,

> We showed that if the distribution used for selecting decoy TXOs are known, then a malicious party can significantly weaken the privacy features of CryptoNote by tracing back the transactions. However, our studies on Monero, a CryptoNote-based blockchain, show that users are using different source codes each having a different algorithms for mixin selection. Fortunately, since there is no distribution that a strong majority of users are using for mixin selection, an attacker cannot derive the distribution of truly spent transactions and arrange the second attack on Monero.

The general direction of the intuition of existing Monero research is correct: statistical identification of nonparametric mixture models can be exceedingly difficult. But ultimately their conclusion is incorrect. Identification is possible by leveraging the structure of Monero's ring signatures.

Each given Monero ring is constructed from a set of decoys independently selected from a particular distribution by the user's wallet software. In the statistical literature, such a structure is known as "repeated

measure(ment)s" [Crowder & Hand, 1990]. Sometimes the concept appears under the labels "exchangeable sequences", "grouped observations", "clustering with instance-level (must-link) constraints", "mixture of product distributions", "multivariate mixtures under the conditionally i.i.d. assumption" "latent-class models" and "latent structure" ([Allman et al., 2009], [Benaglia et al., 2009b], [Ritchie et al., 2020], [Wei & Nguyen, 2022]). A ring's repeated measurements structure provides the critical ingredient for identification and then consistent estimation of the real spend age distribution.[Bonhomme et al., 2016] explains:

> Mixture models are most often parametrically specified, and estimation may be done by maximum likelihood or indirect inference in a frequentist setting, or via Markov chain Monte Carlo techniques in a Bayesian approach.
>
> There is a growing literature on non-parametric identification of finite mixtures. Univariate mixtures are generally not identified non-parametrically. (Additional restrictions may lead to identification. Bordes et al. (2006) and Hunter et al. (2007) studied two- and three-component mixtures of symmetric location families. Kitamura (2004) and Henry et al. (2013) studied models with conditioning variables.) In contrast, data on multiple measurements can represent a powerful source of identification. Hettmansperger and Thomas (2000), Hall and Zhou (2003) and Allman et al. (2009) provided identification results on component distributions and mixing proportions under the assumption that measurements are conditionally independent and the number of components is known. Hu (2008) and Kasahara and Shimotsu (2009) established related results in econometrics.

8 Modeling the Real Spend and Decoy Distributions as a Mixture

I will proceed with the notation of [Bonhomme et al., 2016] and then modify their basic model. Let x be a random variable with support $\mathcal{X} \subseteq \mathbb{R}$. For Monero, the support is block age going back to the first RingCT outputs. Let repeated independent and identically distributed measurements $x_1, x_2, \ldots, x_M$ of x have joint cumulative distribution function (CDF)

$$F(x_1, x_2, \ldots, x_M) = \sum_{k=1}^{K} \omega_k \left(\prod_{m=1}^{M} F_k(x_m) \right) \tag{4}$$

where K is the number of decoy selection algorithms (DSA), ω_k is the proportion of Monero transactions using the kth DSA, M is the number of decoys, and F_k is the CDF of the kth DSA. Of course, each ring also contains the real spend. Let x_S be the random variable representing the age of the real spend. Redefine F_k as $F_{k,D}$ as the CDF of the kth DSA. Then the age of ring members have joint CDF

$$F(x_1, x_2, \ldots, x_M, x_S) = \sum_{k=1}^{K} \omega_k \left(F_{k,S}(x_S) \times \prod_{m=1}^{M} F_{k,D}(x_m) \right) \tag{5}$$

where $F_{k,S}$ is the real spend age distribution within rings that are constructed by the kth DSA. The $F_{k,S}(x_S)$ can be multiplied by $\prod_{m=1}^{M} F_{k,D}(x_m)$ since the decoys are selected independently from the real spend. Here I allow for the possibility — indeed, likelihood — that $F_{k,S}$ is distinct for each DSA. This would occur if certain types of users were more likely to use wallet software with certain DSAs. Spending habits of light wallet users are likely different from users using the Monero GUI (i.e. standard) DSA. If services or centralized exchanges use custom DSAs, their true spend age distribution would also likely be distinct.

In taking the next step there is a small theoretical problem. I aim to transform formula (5) back into the form of (4). This requires that $\prod_{m=1}^{M+1} F_k(x_m) \stackrel{?}{=} F_{k,S}(x_S) \times \prod_{m=1}^{M} F_{k,D}(x_m)$. If $F_k(x_m) = \frac{1}{M+1} F_{k,S}(x_S) + \frac{M}{M+1} F_{k,D}(x_m)$ then a draw from F_k can be modeled as first drawing a Bernoulli random variable with probability of success $p = \frac{1}{M+1}$ ([McLachlan et al., 2019]). If the Bernoulli draw results in a success then draw from $F_{k,S}$. Otherwise, draw from $F_{k,D}$.

The $\prod_{m=1}^{M+1} F_k(x_m) \stackrel{?}{=} F_{k,S}(x_S) \times \prod_{m=1}^{M} F_{k,D}(x_m)$ equality supposes that for every $M+1$ draws from F_k there will be exactly one draw from $F_{k,S}$ and M draws from $F_{k,D}$. In the $\prod_{m=1}^{M+1} F_k(x_m)$ expression there could also be zero draws of $F_{k,S}$, two draws of $F_{k,S}$, three, and so forth. Therefore, $F_{k,S}(x_S) \times \prod_{m=1}^{M} F_{k,D}(x_m)$ is more "exact", "strict", or "restricted" than $\prod_{m=1}^{M+1} F_k(x_m)$. The number of draws from $F_{k,S}$ can be modeled as a binomial distribution with number of draws $n = M+1$ and success probability $p = \frac{1}{M+1}$ ([McLachlan et al., 2019]). When $M = 15$, i.e. rings of size 16, there will be exactly a single draw from $F_{k,S}$ with 38 percent probability. The mean, median, and mode of $Binomial(M+1, \frac{1}{M+1})$ is 1 for all values of $M \geq 0$. I do not think that this theoretical problem causes a significant practical problem. Therefore, I will proceed as if $\prod_{m=1}^{M+1} F_k(x_m) = F_{k,S}(x_S) \times \prod_{m=1}^{M} F_{k,D}(x_m)$.

The model that OSPEAD seeks to estimate is:

$$F(x_1, x_2, \ldots, x_M, x_S) = \sum_{k=1}^{K} \omega_k \left(\prod_{m \in \{1,2,\ldots,M,S\}} F_k(x_m) \right)$$
$$\text{s.t. } F_k(x_m) = \tfrac{1}{M+1} F_{k,S}(x_S) + \tfrac{M}{M+1} F_{k,D}(x_m) \tag{6}$$

The form of the model suggests a two-step procedure: first estimate $F(x_1, x_2, \ldots, x_M, x_S)$ and second estimate the decomposed form of $F_k(x_m)$. To estimate $F(x_1, x_2, \ldots, x_M, x_S)$, a good or practical guess of K, the number of DSAs being used to construct Monero transactions, is necessary. If K is under-estimated then likely some minority DSAs will be grouped together as the same kth DSA. Such a miscellaneous grouping is not necessary to identify the real spend age distribution of the bulk of Monero users, as I will argue in a moment. The first estimation step will produce good estimates of ω_k and $F_k(x_m)$ for each k.

For the second step in which $F_k(x_m)$ is estimated, the mixing proportions $\frac{1}{M+1}$and $\frac{M}{M+1}$ are known since the number of decoys M is known. For identification and estimation of $F_{k,S}$ for a particular k it is necessary to have some knowledge of $F_{k,D}$ for the corresponding k. Fortunately, for several of the $F_{k,D}$ with large ω_k, i.e. the F_k that are most prevalent in the empirical on-chain data, it is possible to obtain an exact form of $F_{k,D}$ since they are described in open source code of Monero wallet software implementations. For the $F_{k,D}$ DSAs where there is not available open source code, my recommendation is to ignore them since identification of $F_{k,S}$ would be on very shaky ground.

The "standard" DSA in `wallet2` that the GUI wallet and many third-party wallets use may be the only relevant $F_{k,D}$. This is a judgment call that the panel should give input on. Since the purpose of OSPEAD is to replace the standard DSA, its protection can really only extend to users who are using the standard DSA. The objective is to set $F_{k,D}(x_m)$ as close to $F_{k,S}(x_S)$ as possible for any particular k.

An alternative choice is to consider the real spend age distribution in aggregate as much as possible for some subset of K DSAs that we have an open source reference for. As of now I am aware of only one other open source DSA that differs from the standard one: MyMonero.[3] If the real spend age distribution of users who use `wallet2` and MyMonero is used to create the OSPEAD DSA, then the new OSPEAD DSA would be:

$$\tilde{F}_{OSPEAD,D}(x_m) = \frac{\omega_{wallet2} F_{wallet2,S}(x_m) + \omega_{MyMonero} F_{MyMonero,S}(x_m)}{\omega_{wallet2} + \omega_{MyMonero}} \tag{7}$$

Note that when $F_{wallet2,S}(x_m) \neq F_{MyMonero,S}(x_m)$ this version (7) of the OSPEAD DSA $\tilde{F}_{OSPEAD,D}$ would not offer best protection for `wallet2` users since the best protection occurs when $F_{OSPEAD,D}(x_m) \approx F_{wallet2,S}(x_m)$.

[3] See "Catalogue of Monero decoy selection algorithms" https://github.com/monero-project/research-lab/issues/99

9 Two-Step Estimation

My proposed estimator of the real spend age distribution is a type of two-step estimator. These estimators are used widely in econometrics.[Murphy & Topel, 1985] explain the mechanics of these models and their challenges:

> The estimation of models that contain unobservable, though estimable, variables is now common in several areas of applied econometrics....In these and other econometric studies, unobserved variables are either replaced by their predicted values from an auxiliary statistical model when estimating the relationship of interest, or they are estimated jointly with that model. This article provides simple but theoretically correct procedures for statistical inference in this class of econometric models when joint procedures are infeasible or inappropriate.
>
> There are two main approaches to estimating this type of model. The first, which we will refer to as the two-step (T-S) estimator, will be our main interest in this article. The T-S procedure simply replaces the unobserved components with their estimated or predicted values from the auxiliary model. Critically, in most applications these values are then treated as if they are known for purposes of estimation and inference in the second-step model, which is usually the model of interest. It is well known that T-S procedures yield consistent estimates of second-stage parameters under fairly general conditions. It is also well known that the second-step estimated standard errors and related test statistics based on these procedures are incorrect. In some cases this feature of the T-S estimator is acknowledged (e.g., Topel 1982. Blanchard 1983), but more commonly the problem is ignored, and no correction of second-step test statistics is attempted. (Some authors who use the T-S procedure write as if the problem were merely one of efficiency [i.e. lower variance]. This conclusion is false, as our subsequent analysis will show.)
>
> Following the suggestion of Liederman (1979), the usual alternative to the T-S procedure is to estimate the first- and second-step models via some joint method, such as full information maximum likelihood (FIML). Under appropriate assumptions, FIML procedures yield efficient estimators and asymptotically correct estimates of standard errors. Unfortunately, in many situations FIML estimation is both computationally complex and costly to implement, facts that account in part for its limited use.

To summarize: two-step estimators are typically consistent, but naive computation of the standard errors of the second-stage parameters would be incorrect since the effect of sampling error in the first stage would be ignored. Joint estimation of the two models will provide correct standard errors, although joint estimation comes with its own drawbacks, assuming it is even possible to develop such a joint estimator.

Correct standard errors (i.e. correct confidence intervals or measures of an estimator's precision generally) are vital for scientific hypothesis testing. For OSPEAD, the important thing is that the two-step estimator is consistent. Obtaining estimates of the precision of the estimator is secondary. As long as the variance of the estimator is not uselessly enormous, using the estimator is preferable to using the log-gamma estimate from the pre-RingCT era based on the analysis of [Möser et al., 2018]. Correct standard errors would be of greatest interest to an attacker aiming to de-anonymize Monero users by leveraging an estimated real spend age distribution as input to a statistical attack. The attacker may want to use hypothesis testing to determine the false positive rate of such an attack. [Liu et al., 2019] provides a framework for statistical privacy based on hypothesis testing. I will return to the question of estimating standard errors in **Section 12 Confidence Interval Estimation**.

10 First Step: Bonhomme-Jochmans-Robin Estimator

The choice of estimator for the first step, i.e. the estimator of ω_k and $F_k(x_m)$ of equation (6), is probably the single most important decision to be made. In every other step there is a way to judge the correctness of the estimates in one way or another. For the first step there is no "ground truth." If we had a reliable ground truth, as [Möser et al., 2018] did, there would hardly be a need to perform statistics in the first place.

There are several methods to consistently estimate (4). The asymptotic variance of most of the existing methods has not yet been determined. The consistency of several methods has not even been rigorously established. They

also tend to be very computationally expensive. One exception is the method proposed in [Bonhomme et al., 2016]. They explain the state of the field:

> To estimate multivariate mixtures non-parametrically, computational procedures akin to the [Expectation Maximization] algorithm (Dempster et al., 1977) have recently been introduced by Benaglia et al. (2009) and Levine et al. (2011). These approaches are applicable more generally than are the earlier proposals of Hettmansperger and Thomas (2000) and Elmore et al. (2004), and of Hall and Zhou (2003) and Hall et al. (2005). Although simulation evidence suggests that these estimators work well in finite samples, their statistical properties are currently unknown. Chauveau et al. (2014) have provided an account of these developments.

[Benaglia et al., 2009b] describe the earlier "cutpoint" methods:

> Hettmansperger and Thomas (2000); Cruz-Medina, Hettmansperger, and Thomas (2004); and Elmore, Hettmansperger, and Thomas (2004) treat the case [that is equivalent to Monero's setting]....
>
> The authors named above have developed an estimation method for the conditionally i.i.d. model. This method, the *cutpoint approach*, discretizes the continuous measurements by replacing each r-dimensional observation, say $\mathbf{X}_i = (x_{i1,} \ldots, x_{ir})$, by the p-dimensional multinomial vector $(n_1, \ldots, n_p)$, where $p \geq 2$ is chosen by the experimenter along with a set of cutpoints $-\infty = c_0 < c_1 < \cdots < c_p = \infty$, so that for $a = 1, \ldots, p$,
>
> $$n_a = \sum_{k=1}^{r} I\{c_{a-1} < x_{ik} \leq c_a\}.$$
>
> Note that the multinomial distribution is guaranteed by the conditional i.i.d. assumption, and the multinomial probability of the ath category is equal to $\theta_a \equiv P(c_{a-1} < X_{ik} \leq c_1)$.
>
> The cutpoint approach is completely general in the sense that it can be applied to any number of components m and any number of repeated measures r, just as long as $r \geq 2m - 1$, a condition that guarantees identifiability (Elmore and Wang 2003). However, some information is lost in the discretization step, and for this reason it becomes difficult to obtain density estimates of the component densities.

The $r \geq 2m - 1$ identifiability requirement for cutpoint methods is probably not very restrictive for our problem. For rings of size 11, the maximum number K of allowable density components F_k to be estimated would be 6. With ring size 16, it would be 8. The main problem with cutpoint methods for us, as [Benaglia et al., 2009b] say, is that the method destroys some information that is necessary to precisely estimate the density components F_k. The primary objective in this first step is to estimate F_k so it can be passed to the second step. Therefore, cutpoint methods are probably a dead end for us.

After cutpoint methods, Expectation Maximization (EM) methods were developed. EM algorithms are often computationally expensive ([O'Hagan et al., 2012]). Some of the papers mentioned above attempt to support the conjecture that their method is consistent through Monte Carlo simulations. For example, [Benaglia et al., 2009a] say,

> Finally, we include a simulation study whose purpose is to explore the possible rate of convergence for the algorithm, for fixed m [number of distribution components] and r [number of repeated measurements], as the sample size n [corresponds to number of rings in Monero] tends to infinity. Note that the plots here do not constitute a proof of the asymptotic rate of convergence, nor even of consistency, yet they are interesting nonetheless because they suggest that such a theoretical result is possible.

Monte Carlo simulations can be helpful in ruling out obvious inconsistency of an estimator, but at best they occupy a distant second place compared to rigorous mathematical proofs. The main issue is that Monte Carlo simulation results are depended upon the particular distributions and parameters chosen for the simulations. Other data

samples may have different properties, resulting in possible inconsistency. Mathematical proofs generally cover all possible cases. It is much safer to go with an estimator that has a rigorous mathematical proof of consistency.

The Bonhomme-Jochmans-Robin estimator described in [Bonhomme et al., 2016] is the complete package: An estimator for an identifiable model that is mathematical proven to be consistent and asymptotically normal, with closed-form expressions for its asymptotic variance and bias. The estimation algorithm is reasonably computationally efficient and the paper authors released a MATLAB implementation. For space reasons I will not fully describe the estimator; the reader is referred to [Bonhomme et al., 2016] itself for full details. I will discuss a few of its features.

10.1 Linear Independence Assumption

Like any somewhat complicated estimator, the Bonhomme-Jochmans-Robin estimator requires that certain assumptions be met. Most of the required assumption are regularity conditions that are easily satisfied in practical applications. The single assumption that is more strict than a simple regularity condition is their Assumption 1:

> *Assumption 1* (rank). The matrix B has maximal column rank.
>
> Assumption 1 is similar to the identification condition of [Allman et al., 2009], theorem 8, which require component distributions to be linearly independent. Indeed, with $\chi_i(x_m) = \mathbf{1}\{x_m \leqslant v_i\}$ for a set of chosen values $v_1, \ldots, v_I$ we have $\mathbb{E}[\chi_i(x_m)] = F_k(v_i)$ and assumption 1 demands linear independence of the component distributions on the grid $v_1, \ldots, v_I$. Note that assumption 1 is testable. Indeed,
>
> $$A \equiv \mathbb{E}[A(x_m)] = B\Omega B'.$$
>
> Hence, for given I, assumption 1 is equivalent to A having rank K, which is a testable restriction. See the on-line supplementary material for details.

Theorem 8 of [Allman et al., 2009] requires that no component distribution can be completely reconstructed by a linear combination of the other component distributions. This would require reconstruction at every point x in the support of such a component. Note that this meaning of linear independence is different from the meaning discussed in **Section 6.5 Identifiability and Identification** with OLS estimation. As argued in [Mbakop, 2017] and [Kwon & Mbakop, 2021], this linear independence assumption of Theorem 8 of [Allman et al., 2009] is generally easily satisfied when the component distributions are continuous.

The story is a bit different with Assumption 1 of the Bonhomme-Jochmans-Robin estimator because the component distributions are evaluated at a discrete number of points even if the distributions are continuous, which could make the assumption harder to satisfy. Using a larger number of points, i.e. setting I to be larger, would probably increase the likelihood that this assumption is satisfied. On the bright side, this Assumption 1 is testable. To give context, it is always very good when a statistical assumption is testable with the sample. When an important assumption is not testable by data, we are left wondering if our estimate is valid without any way to directly test it. OSPEAD will include a statistical test of this Assumption 1 condition.

10.2 Permutation of Triples

The core of the Bonhomme-Jochmans-Robin Estimator involves using every permutation (*not* combination) of ordered triples (i.e. 3-tuples) from each set of repeated measures. Certain characteristics of the estimator flow from the triple permutations. First, the minimum number of repeated measurements necessary for identification and estimation is 3. That presents no problems for us since ring size is now 16, up from the pre-hardfork 11. As will be discussed in detail in **Section 12 Confidence Interval Estimation**, the variance of the estimator shrinks in rough relation to n choose k, i.e. the binomial coefficient $\binom{n}{k} = \frac{n!}{k!(n-k)!}$, where n is the ring size and $k = 3$.

The triple permutations also influence the computational expense of the estimator. By taking triples rather than dealing with the entire set of repeated measures, the Bonhomme-Jochmans-Robin estimator skirts the curse of dimensionality that afflicts other estimators of its class. The main computational bottleneck requires computation and summation of $N \times I^3 \times \frac{(M+1)!}{(M+1-3)!}$ terms, where N is the number of "rings", M is the number of decoys to maintain the same notation as above (And thus $M+1$ is ring size), and I is the number of "evaluation points" of the distributions. I have verified this quantity through code profiling of the paper's MATLAB code. The choice of I is important for precision of the estimator. Larger I increases precision. Table 1 of [Bonhomme et al., 2016] gives the Monte Carlo bias and standard errors of the estimates of the means of the component distributions with $I = 5$ and $I = 10$. With greater I there is a moderate improvement in precision. Computational expense is the limitation on I because expense increases with the cube of I. Therefore, it would be best to convert the MATLAB implementation into a C++ implementation, which I discuss in **Section 18 OSPEAD Requests to Monero C++ Developer(s)**, so that I can be made as large as possible.

10.3 Determining the Number of Distribution Components K

Historically, determining the number of distribution components K for fitting finite mixture models has relied on heuristics with questionable foundations, especially in the nonparametric setting [McLachlan & Peel, 2000]. Usually, K must be specified at the very beginning of the estimation procedure and therefore OSPEAD would be incomplete without a specification of how to choose K. In our setting, the modeling choice or estimate of K is equivalent to the choice or estimate of the number of decoy selection algorithms being used in the wild on the blockchain. Most papers on estimating mixture models consider the choice of K to be outside of their scope, and [Bonhomme et al., 2016] is no exception. Fortunately, in the last decade a few papers have made progress.

[Kasahara & Shimotsu, 2014] analyze the case of multivariate mixtures where the multiple variables "are independently (but not necessarily identically) distributed within each component." Our repeated measures is a special case of this multivariate setting: the variables are both independently and identically distributed within each component (or at least identically distributed in the sense of Equation (6)). Therefore, the [Kasahara & Shimotsu, 2014] method would also work in our stricter setting. [Kasahara & Shimotsu, 2014] give an identification result when there are at least two variables (in our setting, at least two ring members) as well as a procedure to consistently estimate a lower bound on the number of distribution components.

[Kwon & Mbakop, 2021] improves on [Kasahara & Shimotsu, 2014] by creating a consistent estimator of the exact number of components (not just a lower bound). From their Monte Carlo simulations it appears that their estimator is not very precise when K is high but N (number of rings) is low. In our case we have both K and N high, so their estimator should correctly estimate the number of components. Therefore, as of now I plan to use their procedure to estimate K despite the fact that there are two complications. The first is that [Kwon & Mbakop, 2021] do not make their code available. Either I could re-implement the method or we could ask them if they are willing to share the code. The second complication is that the method requires computing each element of two $N \times N$ matrices, solving for their matrix square root and then computing the singular value decomposition of their product (see Algorithm 1 of [Kwon & Mbakop, 2021]). Given that our target N is about $400,000$, [Kwon & Mbakop, 2021] could get computationally expensive. Taking a random subsample may be necessary.

10.4 Comparison with Ritchie-Vandermeulen-Scott Estimator

I will now compare the Bonhomme-Jochmans-Robin estimator to an estimator described in [Ritchie et al., 2020], which is based on theoretical results from [Vandermeulen & Scott, 2019]. At one point I considered the [Ritchie et al., 2020] estimator to be the leading candidate for a potential estimator for the first step, but on deeper analysis it is clear to me that it is inferior to the Bonhomme-Jochmans-Robin estimator for our particular problem.

Overall, [Ritchie et al., 2020] approaches the problem from more of a machine learning angle. (Bonhomme, Jochmans, and Robin are econometricians.) They show identifiability and consistency of their estimator, but do not give an expression for its variance, unlike [Bonhomme et al., 2016]. They are particularly concerned with classification of observations into the component distributions after estimation and compare their estimator with existing clustering methods in the vein of machine learning.

[Ritchie et al., 2020] and [Vandermeulen & Scott, 2019] make some theoretical achievements such as showing that a mixture distribution with repeated measurements is identifiable even if the number of repeated measurements is only two as long as an additional "joint irreducibility" condition is met. [Bonhomme et al., 2016] requires at least three. This achievement is not relevant for our problem since our number of repeated measurements is 11 and 16. The method of [Ritchie et al., 2020] also can be used with multiple variables, unlike that of [Bonhomme et al., 2016]. This is not relevant to us since the age of spent transaction outputs is univariate.

As an aside, the [Ritchie et al., 2020] describe why conventional clustering methods cannot identify nonparametric mixture models with repeated measurements:

> Mixture models are often utilized to solve the clustering problem. Parametric mixture models, such as GMMs, are able to capture overlapping clusters. Most clustering algorithms, however, such as k-means [11, 12], DBSCAN [13], and spectral clustering [14, 15], assume clusters are non-overlapping and hence fail when clusters overlap.

By "overlapping clusters" here they mean settings where the component distributions F_k have partial or completely overlapping support (such the setting of Monero's ring signature age distribution). See [Vankadara et al., 2021] for more discussion of this point.

The computational expense of [Ritchie et al., 2020] basically torpedoes its use in our setting, unfortunately. The Python implementation code and the example data all used two repeated measures. It took me some time to realize why: the computational expense of their estimator explodes as the number of repeated measurements increases. According to my reading of the paper and the included Python code, the number of terms that need to be summed is approximately $N \times I^{M+1}$ where N is the number of "rings", M is the number of decoys, and I is the number of "evaluation points" of the distributions as before. The meaning of evaluation points is different in [Ritchie et al., 2020], but it serves a similar roles as in [Bonhomme et al., 2016]. A minimum acceptable value of I is probably about 10. If $N = 100,000$, $I = 10$, and $M = 15$, then the number of terms to sum would be $100,000 \times 10^{16} = 10^{21}$, which is computationally infeasible. With Bonhomme-Jochmans-Robin estimator, the required number of terms to sum would be only $100,000 \times 10^3 \times \frac{16!}{13!} = 3.36 \times 10^{11}$. (With ring size 128, the estimator would require summation of $100,000 \times 10^3 \times \frac{128!}{125!} = 2.03 \times 10^{14}$ terms, which seems computationally feasible with an optimized implementation.)

One way to reduce the computational expense of the [Ritchie et al., 2020] estimator is to take a random subsample of each ring, e.g. only select 4 ring members from each ring. In theory there should be no theoretical problem with doing this. The estimator should still be consistent since the sample was selected in an independent way. Such an approach is probably inferior to using the Bonhomme-Jochmans-Robin estimator with the full sample. It would be hard to justify the subsampling given that we do not even know the variance of the [Ritchie et al., 2020] estimator. Why we should accept excluding data when the Bonhomme-Jochmans-Robin estimator uses all observations?

10.5 Options for the First Step Estimator

My recommendation is to use the Bonhomme-Jochmans-Robin Estimator for the first step. Possible alternatives are:

1. The Ritchie-Vandermeulen-Scott Estimator. It is likely inferior to the Bonhomme-Jochmans-Robin Estimator since its variance is unknown and its computational expense is large.

2. The estimator described in [Benaglia et al., 2009a]. The `npEM` function in the `mixtools` R package implements this estimator. It is likely inferior to the Bonhomme-Jochmans-Robin Estimator since its consistency has not yet been proven.

3. The estimator described in [Levine et al., 2011]. The `npMSL` function in the `mixtools` R package implements this estimator. It is likely inferior to the Bonhomme-Jochmans-Robin Estimator since its consistency has not yet been proven.

Another option is to use the [Benaglia et al., 2009a] and/or [Levine et al., 2011] EM estimators as "checks" on the Ritchie-Vandermeulen-Scott Estimator. I do not recommend this because it is a kind of tautology. If those EM estimators arrive at similar estimates as the Ritchie-Vandermeulen-Scott Estimator, then we conclude that the Ritchie-Vandermeulen-Scott Estimator gives a good estimate. If the EM estimators arrive at a dramatically different estimate (i.e. well outside of the confidence intervals — this can be made formal with a hypothesis test and joint bootstrapping), then what should we conclude? It seems obvious to trust the estimate of a consistent estimator rather than the estimate of estimators with unknown consistency properties. We use the Ritchie-Vandermeulen-Scott Estimator anyway.

The number of distribution components K must also be decided. I recommend the estimator described in [Kwon & Mbakop, 2021]. Possible alternatives are:

1. Use the [Kasahara & Shimotsu, 2014] estimator to estimate a lower bound on K. Then choose some K greater than or equal to that estimated value.

2. Use another method with poorer theoretical foundations such as sequential hypothesis testing (SHT), Akaike Information Criteria (AIC), Bayesian Information Criteria (BIC), and Hannan-Quinn Information Criteria (HQ), which both [Kasahara & Shimotsu, 2014] and [Kwon & Mbakop, 2021] compared their proposed estimators against.

3. Select K based on "expert knowledge" of the probable number of decoy selection algorithms being used in the wild.

11 Second Step: Patra-Sen Inversion Estimator

Once each $F_k(x_m)$ from model (6) is estimated, the real spend age distributions $F_{k,S}(x_S)$ of the each component of interest must be estimated. [Patra & Sen, 2016] describe a model and estimator that can be used to estimate $F_{k,S}(x_S)$. They begin their paper by setting up their main model of interest:

> Consider a mixture model with two components, i.e.
>
> $$F(x) = \alpha F_{\mathrm{s}}(x) + (1-\alpha)F_{\mathrm{b}}(x) \tag{8}$$
>
> where the cumulative distribution function (CDF) F_{b} is known, but the mixing proportion $\alpha \in [0,1]$ and the CDF $F_{\mathrm{s}}\,(\neq F_{\mathrm{b}})$ are unknown. Given a random sample from F, we wish to estimate (non-parametrically) F_{s} and the parameter α....
>
> In this paper we provide a methodology to estimate α and F_{s} (non-parametrically), without assuming any constraint on the form of F_{s}.

The paper briefly discusses the model that corresponds to our problem: when α — the proportion of ring members that are real spends — is known:

Suppose that we observe an independent and identically distributed sample $X_1, X_2, \ldots, X_n$ from F as in model (8). If $\alpha \in (0, 1]$ were known, a naive estimator of F_{s} would be

$$\hat{F}^{\alpha}_{\mathrm{s},n} = \frac{\mathbb{F}_n - (1-\alpha)F_{\mathrm{b}}}{\alpha} \tag{9}$$

where $\mathbb{F}_n$ is the empirical CDF of the observed sample, i.e. $\mathbb{F}_n(x) = \sum_{i=1}^{n} \mathbf{1}\{X_i \leqslant x\}/n$. Although this estimator is consistent, it does not satisfy the basic requirement of a CDF: $\hat{F}^{\alpha}_{\mathrm{s},n}$ need not be non-decreasing or lie between 0 and 1. This naive estimator can be improved by imposing the known shape constraint of monotonicity. This can be accomplished by minimizing

$$\int \left\{W(x) - \hat{F}^{\alpha}_{\mathrm{s},n}(x)\right\}^2 \mathrm{d}\mathbb{F}_n(x) \equiv \frac{1}{n}\sum_{i=1}^{n} \left\{W(X_i) - \hat{F}^{\alpha}_{\mathrm{s},n}(X_i)\right\}^2 \tag{10}$$

over all CDFs W. Let $\check{F}^{\alpha}_{\mathrm{s},n}(x)$ be a CDF that minimizes expression (10).

Note that $\mathbf{1}\{x\}$ is the indicator function.[4] Patra and Sen then describe a numerical algorithm to minimize (10) with computational complexity $O(n)$. Equation (9) has been referred to as an "inversion estimator" [Milhaud et al., 2021]. It is a type of plug-in estimator because estimated quantities (here, only $\mathbb{F}_n$ needs to be estimated since α and F_b are known) are plugged into the equation to determine the estimate. Their equation (9) is very similar to equation (2) in my HackerOne submission. [Patra & Sen, 2016] used a CDF form. I used a PDF form. The definitions of α and $(1-\alpha)$ are switched in the two papers. I also recognized the issue of constructing a valid probability distribution and chose a less sophisticated correction:

> Note that due to the long thin tails of the distributions and the noisy nature of the empirical data and jberman's simulated data, it can be the case that the calculated value of $f_S(x)$ [the PDF of the real spend age distribution] can be negative for some values of x, which violates an axiom of probability theory. In those cases, the negative value of $f_S(x)$ was replaced by $f_M(x)$ [the PDF of the decoy selection algorithm], since that would be the most conservative approach.

Patra and Sen's (9) is also similar to equation (2) in [Aeeneh et al., 2021], which develops the MAP Decoder attack on Monero, although in that paper it is described as a probability relationship rather than as a plug-in estimator per se.

We are quite fortunate that the structure of ring signatures guarantees that α is known with certainty rather than being a parameter that must be estimated. The value of α is not just known in the population, i.e. *in expectation*, but is known to be exactly $\frac{1}{M+1}$ in the sample. [Patra & Sen, 2016] note, "When α is unknown, the problem is considerably more difficult; in fact, it is non-identifiable." This mixture model is sometimes referred to as a contamination or admixture model [Milhaud et al., 2021].

[Patra & Sen, 2016] does not derive the variance of $\check{F}^{\alpha}_{\mathrm{s},n}$. There are a few papers with similar, more general models (such as treating α as unknown) that have some variance estimator. [Arias-Castro & Jiang, 2021] use a bootstrap procedure, for example.

11.1 Options for the Second Step Estimator

The equation (8) has four unknowns to start with: $F(x)$, α, $F_{\mathrm{s}}(x)$, and $F_{\mathrm{b}}(x)$. When two of them (α and $F_{\mathrm{b}}(x)$) are known with certainty and one of them ($F(x)$) can be estimated, we have one linear equation in one unknown. Therefore, it seems that the Patra-Sen Inversion Estimator is very suitable. I am not aware of an alternative estimator. There is not an obvious way to improve on the estimator, except perhaps in the rectification step of equation (10). The rectification of $\hat{F}^{\alpha}_{\mathrm{s},n}$ into a valid CDF is not just a matter of theoretical tidiness. $\hat{F}^{\alpha}_{\mathrm{s},n}$ needs

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

to be passed to yet another step to fit a parametric distribution over an estimate of a valid PDF. Therefore, some rectified estimate such as $\check{F}^{\alpha}_{\mathrm{s},n}(x)$ is necessary. In saying that the rectification step could possibly be improved, I do not mean that my ad hoc rectification that I performed in my HackerOne submission is an improvement. Rather, it is possible that there exists a published rectification method that is superior to the Patra-Sen method for our particular setting. I am not aware of any. The options are thus:

1. Use the Patra-Sen Inversion Estimator, including the rectification in Equation (10).
2. Use the Patra-Sen Inversion Estimator, but use the ad hoc rectification that I used in my HackerOne submission.

I recommend option (1).

I will state here an alternative to the two-step estimation procedure I have recommended. We could just ignore the existence of other decoy selection algorithms and apply the Patra-Sen Inversion Estimator directly to the data without using the first step. This is roughly equivalent to what I did in my HackerOne submission. Obviously, I do not recommend this, but I wanted to raise it as an option. The performance of such an estimator would worsen with the ring size increase from 11 to 16 because there would be even more "noise" in the raw data from the nonstandard decoy selection algorithms compared to the signal of real spends.

12 Confidence Interval Estimation

As I stated before, for our current problem the estimation of confidence intervals is secondary to the main goal of consistently estimating the real spend age distribution. It is still useful to estimate confidence intervals since excessively large confidence intervals might suggest that sample size should increase — e.g. increase the time window from one week to one month.

Given time constraints and the technical difficulty of deriving a closed-form expression for the two-step estimator outlined above, bootstrapping is the obvious method for estimating confidence intervals.

An interesting object of investigation is how the width of confidence intervals respond to a change in ring size. ArticMine has taken a particular interest in this question. A larger ring size gives the Bonhomme-Jochmans-Robin estimator more information to estimate each component F_k in the first step. Therefore, raising ring size will shrink confidence intervals in the first step. In the second step, we might expect the confidence interval on the estimate $F_{k,S}$ to expand as the signal-to-noise ratio falls when ring size increases. These two effects move in opposite directions, so the direction of their combined effect is a question of their relative magnitudes.

12.1 Variance of the First Step Estimator

[Bonhomme et al., 2016] give the following closed-form expression for the asymptotic distribution of their estimator of the component densities:

Proposition 2. Let assumptions 1–3 and 5 and 6 hold. Then $|\hat{f}_k(x) - f_k(x)| = o_P(1)$ and

$$\sqrt{(Nh)}\left\{\hat{f}_k(x) - f_k(x)\right\} \xrightarrow{L} \mathcal{N}\left(\sqrt{c}\mu_f, \mathscr{V}_f\right),$$

as $N \to \infty$ where $c \geqslant 0$ is a finite constant,

$$\mu_f \equiv \frac{1}{2} f_k^{''}(x) \int_{-\infty}^{\infty} u^2 \kappa(u) \mathrm{d}u,$$

$$\mathscr{V}_f \equiv M \left\{\frac{(M-3)!}{M!}\right\}^2 q_k(x) f(x) \int_{-\infty}^{\infty} \kappa(u)^2 \mathrm{d}u,$$

provided that $Nh \to \infty$ and $Nh^5 \to c$.

In proposition 2 we allow the bandwidth to vanish at the optimal rate of $N^{-1/5}$.

Proposition 2 says that the difference between the true density and the estimator converges to a normal distribution with mean (i.e. bias) $\sqrt{c}\mu_f$ and variance $\mathscr{V}_f$ when the kernel bandwidth is h. As I said in **Section 6.6 Variance of an Estimator**, nonparametric density estimators have a slower convergence rate than parametric estimators. And some bias is generally unavoidable. $f_k^{''}(x)$ in μ_f is the second derivative of the kth component's PDF at point x. The $\kappa(u)$ is the chosen kernel function, which is usually the Normal kernel.

The meaning of $q_k(x)$ in the definition of $\mathscr{V}_f$ is available in [Bonhomme et al., 2016] and not too important for the following discussion. The relevant fact is that $q_k(x)$ does not depend on M. The point I want to make is that for a fixed value of $q_k(x)f(x)\int_{-\infty}^{\infty}\kappa(u)^2\mathrm{d}u$, the estimator's asymptotic variance $\mathscr{V}_f$ is proportional to $M\left\{\frac{(M-3)!}{M!}\right\}^2$. Since the estimator's asymptotic distribution is Normal, it is easy to compute the effect of an increased ring size on the confidence interval when our sample size is very large.

A 90% confidence interval (a reasonable interval for our current problem) would be $\left[\hat{f}_k(x) - 1.64\sqrt{\mathscr{V}_f}, \hat{f}_k(x) + 1.64\sqrt{\mathscr{V}_f}\right]$. In other words, the width of the confidence interval is proportional to the square root of the variance of the estimator, i.e. its standard error. Therefore, to evaluate the effect of ring size on the confidence interval of estimator $\hat{f}_k(x)$, the ratio of the two $\sqrt{\mathscr{V}_f}$ can be calculated. In other words, $\sqrt{M\left\{\frac{(M-3)!}{M!}\right\}^2} = \sqrt{M}\frac{(M-3)!}{M!}$ should be calculated for two different M values. Then the expected ratio of the width of the confidence intervals when ring size is 16 compared to ring size 11 is

$$\left(\sqrt{16}\frac{(16-3)!}{16!}\right) / \left(\sqrt{11}\frac{(11-3)!}{11!}\right) \approx 0.36$$

Therefore, we may expect that the confidence intervals of $\hat{f}_k(x)$ when ring size is 16 would only be 36% the width of confidence intervals when ring size is 11, i.e. confidence interval width would fall by 64 percent. In the broader context, the Flourine Fermi hard fork thus contributed to greatly improved precision *for the first step of the estimator*. If Monero's ring size increases to 128 as is probable with the Seraphis upgrade, then the ratio of the confidence intervals for ring size 16 vs 128 would be

$$\left(\sqrt{128}\frac{(128-3)!}{128!}\right) / \left(\sqrt{16}\frac{(16-3)!}{16!}\right) \approx 0.005$$

So the confidence interval would be less than half a percent in width with ring size 128 compared to 16. In other words, ring size of 128 (without binning) would produce an extremely precise estimate of $f_k(x)$. The computational expense of the estimator would rise, however.

12.2 Variance of the Second Step Estimator

This is only half of the story, of course. The variance of the Patra-Sen Inversion Estimator must be considered as well. As I stated in **Section 11 Second Step: Patra-Sen Inversion Estimator**, I am unaware of any closed-form expression for the asymptotic variance of the Patra-Sen Inversion Estimator. In cases such as these, Monte Carlo simulation usually gives a good approximation of the variance of the estimator, at least for the parameters chosen for the simulation. The `patra-sen-monte-carlo.R` code for a simulation is included in this zip file.

For this simulation I set the real spend age distribution to be negative binomial with mean parameter 100 and dispersion parameter 3. Then the decoy distribution was negative binomial with mean parameter 150 and dispersion parameter 2. 100,000 rings were constructed for each of 1,000 Monte Carlo simulations, with ring size 11, 16, and 128. I applied the Patra-Sen Inversion Estimator with rectification $\check{F}_{\mathrm{s},n}^{\alpha}(x)$ in each simulation (plus a kernel smoothing step) and obtained a 90 percent confidence interval at each point in the support x by calculating the 5th and 95th percentile of the simulated estimates. I chose confidence intervals instead of standard errors as

the metric of comparison since it is unknown at this point in time whether $\check{F}^{\alpha}_{\mathrm{s},n}(x)$ is asymptotically normal — and therefore it is unknown whether the width of the confidence intervals are strictly proportional to the standard errors. I plotted these confidence intervals and split the support to show detail in Figure 1.

The first thing to notice in these plots is that the confidence intervals when ring size is 16 are only slightly larger than when ring size is 11. I calculated the median of the width of all confidence intervals when $x \in \{20, 21, \ldots, 200\}$ for ring size 11, 16, and 128 to generate a summary statistic for comparison. According to this median, the confidence interval for ring size 16 was only 14 percent larger than for ring size 11. However, precision suffered quite a bit when ring size rose to 128: the confidence interval with ring size 128 was 212 percent wider than with ring size 16.

Another thing to notice is the estimator's apparent bias in the support range 1 to 20. As I have mentioned before, some bias is generally unavoidable with nonparametric kernel density estimators. Looking carefully at the bias here, it is notable that the estimator is *downward* biased when the decoy distribution is greater than the real spend age distribution. My intuition tells me that the bias probably is not being caused by the density of decoy distribution; if it were, I would expect an upward — not downward — bias in this range of the support. The bias may be due to the typical bias with kernel smoothing or some effect of the imposing the monotonicity constraint on the CDF as in equation (10). In general, bias correction procedures are possible, but in practice they are often avoided since they usually increase the variance of estimators too much.

Despite the analysis above, we still cannot say with certainty whether a rise in ring size from 11 to 16 increases or decreases the precision of my proposed two-step estimator. This is for two reasons. First, the confidence interval of the first step's estimator may shrink by 64 percent and the confidence interval of the second step's estimator may expand by only 14 percent, but we cannot know which effect dominates without knowing the initial confidence interval widths for both estimators at ring size 11. If the confidence interval of the first step estimator is very small to begin with and the confidence interval of the second step estimator is very large, then 14 percent of a large number could be much bigger than 64 percent of a small one.

The second reason is the general complication of variance estimation in two-step models discussed in [Murphy & Topel, 1985] and related work. Unfortunately, the answer to this overall question will have to wait for the full bootstrap or Monte Carlo estimations. To be clear, bootstrapping and Monte Carlo estimates for two-step estimators must do the (re)sampling at the first step and then carry that same sample to the second step estimate, then resample again at the first step for the next iteration.

Bootstrap-based confidence intervals involve a special challenge with finite mixture models. [Bonhomme et al., 2016] explains:

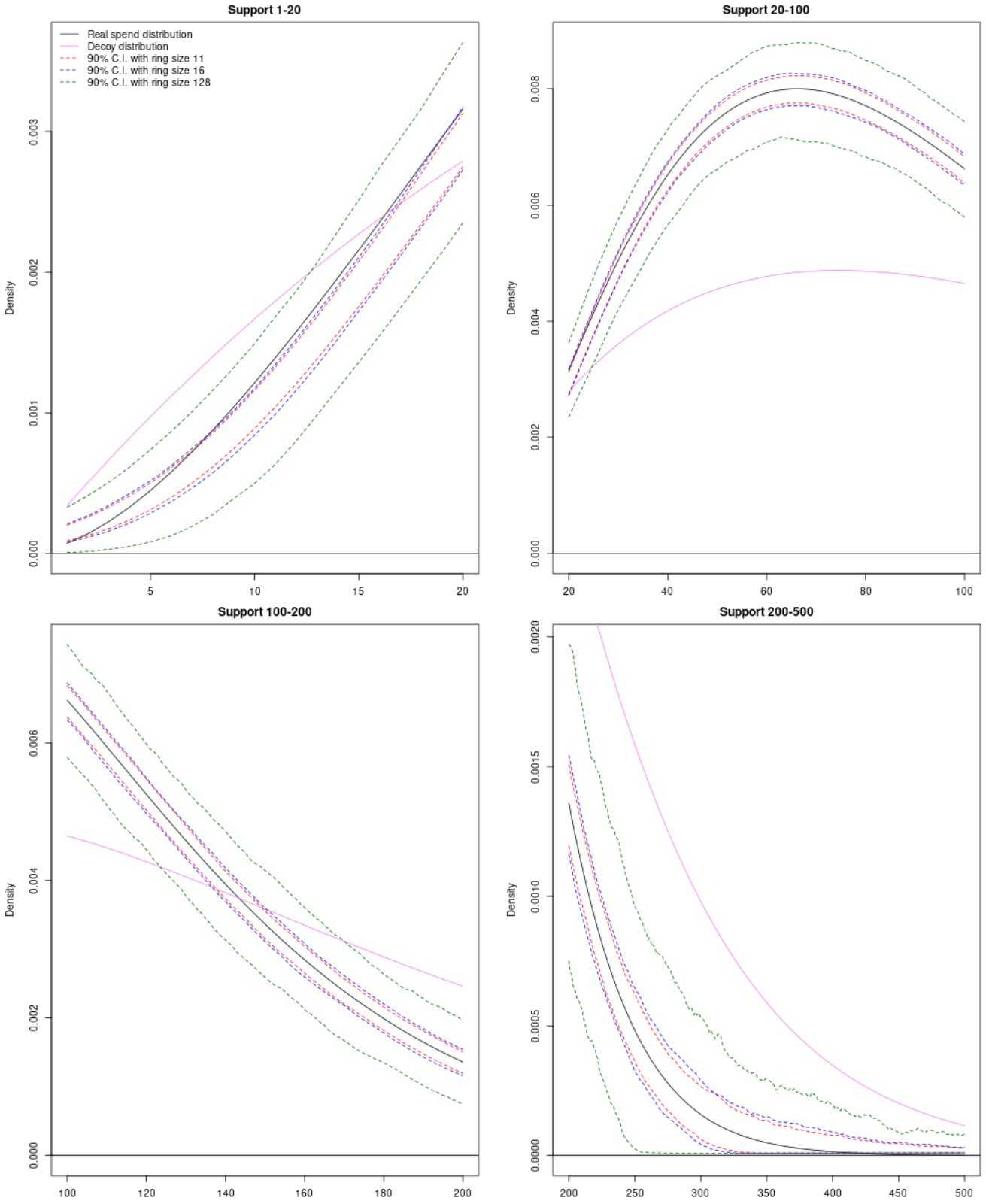
> We note that, although it must be taken care of in our simulations, label swapping does not cause any complications for estimation and inference based on our approach. Label swapping does present an important challenge for inference methods based on resampling algorithms such as the bootstrap or jackknife, and on Markov chain Monte Carlo procedures (see, for example, Stephens (2000)).

We need to be able to say that the kth F_k estimated distribution in the first step is associated with the kth decoy selection algorithm for estimation in the second step. It is perhaps not too difficult to do this when a human can examine a single estimate. Bootstrapping involves many such estimates; I would aim for 1,000 for OSPEAD. Therefore, some automated way of labeling is recommended. [Bonhomme et al., 2016] describe their own method to avoid label-swapping in their Monte Carlo simulations:

> To deal with label swapping in our simulation study we proceed as follows. In each replication we first estimate the mean of each component. We then label the estimated component with the smallest estimated mean as the first mixture component. We carefully checked for label swapping and found no mix-up.

The mean of distributions is not very meaningful nor specific in our setting. Here I tentatively develop two methods that may work alone or combined to perform automated labeling. The first is to simply assume that the mixing

Figure 1: Patra-Sen Monte Carlo Confidence Intervals by Ring Size



proportions ω_k would remain stable in each bootstrap iteration. My hypothesis is that most transactions are being constructed with the standard `wallet2` DSA, followed by the MyMonero DSA as the second most prevalent. We will have to see what the data tells us to confirm or invalidate that hypothesis.

The second possible approach to automated labeling involves enforcement of one of the probability axioms: that probability cannot be negative. The idea is similar to the imposition of monotonicity of the CDF in equation (10) from the Patra-Sen Inversion Estimator. Once a specified DSA distribution is subtracted from the mixture distribution F_k, the remaining portion is the estimate of the real spend age distribution $\hat{F}_{k,S}$ (which is the label we care about). Of course, due to sampling error some portion of the corresponding estimated real spend age distribution $\hat{F}_{k,S}$ may appear as negative; this suggest a statistical test should be developed, or at least a heuristic. As ring size increases, this type of automated labeling should get more accurate since there is less flexibility in the thickness of the real spend age distribution $F_{k,S}$ as a proportion of the component distribution F_k.

12.3 Variance of the Estimator of the Number of Distribution Components K

The estimator for the number of distribution components K is also subject to sampling error. [Kwon & Mbakop, 2021] do not give a way to compute their estimator's variance or confidence intervals. Bootstrapping could be a reasonable strategy because it is usually valid even when the parameter space is discrete [Newton, 1996]. The parameter space of K is the natural numbers. However, my recommendation would be to ignore the variance of $\hat{K}$ and to use the (weekly) estimate of K as its true value because:

1. K is discrete and probably less than 10. Therefore, the practical variance of $\hat{K}$ is probably negligible given our sample sizes.

2. A proper bootstrap procedure for $\hat{K}$ creates special problems for the downstream estimators. Say a bootstrap estimate of K produces varied estimates of K, which anyway would be the whole point of computing a bootstrap iteration: to estimate the variability of $\hat{K}$. Then some of the downstream bootstrap estimates of the Bonhomme-Jochmans-Robin Estimator and the Patra-Sen Inversion Estimator would have a varied number of estimated components. Then combining those estimates for bootstrap confidence intervals would result in some "missing" and "extra" components, greatly complicating valid statistical inference.

3. The estimator described in [Kwon & Mbakop, 2021] is computationally expensive. If bootstrapped, the procedure would have to be done about 1,000 times for each week of data rather than a single time for each week.

12.4 Options for Confidence Interval Estimation

My recommendation is to perform about 1,000 nonparametric bootstrap iterations, starting each iteration from the first step and then passing the results of the first step to the second step in each iteration. A nonparametric bootstrap, also known as a case resampling bootstrap, requires the fewest assumptions about our model and estimator compared to other bootstrap subtypes. Setting the number of iterations to 1,000 should provide enough information to calculate confidence intervals with good coverage.[5] If the computation is very expensive, then we could settle for 100 iterations. Computation of a closed-form asymptotic confidence interval is not really an option due both to the two-step nature of the estimator (as [Murphy & Topel, 1985] warn) and because there is not yet a closed-form expression for the variance of the Patra-Sen Inversion Estimator.

Another option would be to not estimate any confidence intervals and just expect that the consistency properties of the estimator will give us a good estimate. As a statistician, having no metric of precision at all would make me nervous. Granted, the sample size is quite large.

[5] "Coverage" used with this meaning: https://en.wikipedia.org/wiki/Coverage_probability

13 Sampling Strategy and Data Features that Support Estimation

Earlier I said that we have no "ground truth" about the different distribution components F_k. Even with no ground truth, the data has certain features that can serve as a check on the estimation.

The plan is to take each week from September 1st, 2021 to October 31, 2022 as a sample. Monero version 0.17.2.3, released September 1, 2021 was the last time that the standard decoy selection algorithm was changed. Starting the sample in September 2021 would also avoid the apparent July-August 2021 flood incident that distorted the ring member age distribution ([Krawiec-Thayer et al., 2021]).

There is a clear weekly cycle in the on-chain transaction volume data. Taking each week as the estimation unit would avoid the need to deal with the cycles explicitly. The average number of rings (i.e. transaction inputs) per week since September 2021 has been about 375,000, according to the `monero-blockchain-stats` command line utility. This sample size is probably large enough for acceptable precision of my proposed estimator. Two-step estimators tend to have larger variance. The fact that there is a nonparametric kernel estimator involved also raises variance. If 375,000 observations are not enough, then we can take samples 4 weeks at a time, making the sample size 1.5 million rings. It is crucial to break the sample into time units rather than pooling the entire sample together because the real spend age distribution is expected to change over time.

Table 1: Transactions Per Day Around the Hard Fork

Date	Transactions
2022-08-06	21,315
2022-08-07	23.297
2022-08-08	29,083
2022-08-09	27,178
2022-08-10	32,995
2022-08-11	25,566
2022-08-12	61,305
2022-08-13	47,536
2022-08-14	9,288
2022-08-15	18,289
2022-08-16	20,555
2022-08-17	27,398
2022-08-18	29,639
2022-08-19	40,914
2022-08-20	23,031
2022-08-21	20,702

Source: `monero-blockchain-stats`

13.1 Leveraging the Hard Fork

I want to have at least two months of post-hard fork data for a few different reasons. In my OSPEAD CCS proposal I wrote[6]:

> The upcoming hard fork, which does not yet have a fixed date, will include an increase in the ring size. The discontinuity that the hard fork creates can be leveraged to better understand how ring signatures work in pratcice [sic] on the Monero blockchain. Therefore, some of the research work will occur after the hard fork.

First, it is likely that the larger ring size will shrink the confidence intervals of the estimates, as I discussed in **Section 12 Confidence Interval Estimation**. Second, the hard fork caused user and software disruption that could generate "cleaner" estimates. Table 1 lists the daily on-chain transaction volume around the time of the hard fork. The day before the August 13 hard fork, transaction volume rose to about double normal levels. The day after the hard fork, transaction volume plummeted by about 2/3rds.

A large number of wallet implementations did not update in time for the hard fork. Until they were updated, no transactions constructed from those wallets would have appeared on the blockchain. Some of these wallets are known to use non-standard decoy selection algorithms, such as MyMonero, Edge, and Exodus.[7] The decoy selection algorithms of some of the other wallets is unknown. Table 2 contains the approximate dates that some of these wallets could begin constructing valid transactions again. During the period of time that these wallets were out of commission, the proportion of transactions on the blockchain with the standard decoy selection algorithm, i.e. the mixing proportion $\omega_{wallet2}$ in Equation (6), would have been much higher. Therefore, it would likely be easier to estimate $F_{wallet2}$ without the distracting other components F_k during this immediate post-hardfork period.

[6] https://ccs.getmonero.org/proposals/Rucknium-OSPEAD-Fortifying-Monero-Against-Statistical-Attack.html
[7] https://github.com/monero-project/research-lab/issues/99

Table 2: Delayed Hardfork-compatible Wallets as of August 30, 2022

Fix date	Wallet	Source
August 19	WooKey	https://github.com/WooKeyWallet/monero-wallet-android-app/releases/tag/v2.2.1
August 25	Exodus	https://twitter.com/exodus_io/status/1562918301181034496
August 27	Edge	https://twitter.com/EdgeWallet/status/1563584457361149952
August 29	MyMonero	https://twitter.com/MyMonero/status/1564149853478760448
Not fixed	Atomic	https://www.reddit.com/r/atomicwallet/comments/x0w1jr/xmr_balance_showing_zero_i_sent_a_transfer_a_few/
Not fixed	Coinomi	https://www.reddit.com/r/COINOMI/comments/wrqvw7/monero_node_is_down_need_to_export_the_mnemonic/
Not fixed	Guarda	https://libera.monerologs.net/monero-dev/20220831#c143060
Not fixed	Zelcore	https://twitter.com/CactusHashAZ/status/1564020934482178048#m

13.2 MyMonero Fee Fingerprinting

jberman discovered that the MyMonero wallet used a method to calculate transaction fees that is different from `wallet2`'s method.[8] jberman concluded, "Thus, this difference from wallet2 should be fingerprintable on chain." MyMonero also uses a distinct decoy selection algorithm.[9] The fee fingerprint can be used in two ways. First, the proportion of MyMonero transactions $\omega_{MyMonero}$ in Equation (6) can be calculated exactly for each week rather than estimated with the Bonhomme-Jochmans-Robin estimator. This can serve as a "ground truth" check on the estimated $\hat{\omega}_{MyMonero}$ that would hopefully reveal any problems with the Bonhomme-Jochmans-Robin estimator in this setting.

The second way that the MyMonero fee fingerprinting could be useful is in partitioning the sample. If the MyMonero component of the mixture distribution is completely excluded from the first step estimate, then a certain source of noise is removed. We could exclude these MyMonero transactions and be done with it if we wanted to construct the OSPEAD decoy selection algorithm based only on the real spend age distribution of users of the `wallet2` DSA. If we wanted to use the combined `wallet2` and MyMonero real spend age distribution as envisaged in equation (7), then we could still exclude MyMonero from the main estimation, but then take the MyMonero-only sample and apply the Patra-Sen Inversion Estimator to it directly, skipping the first step Bonhomme-Jochmans-Robin estimator.

13.3 Nonstandard Wallets are Nonstandard in Multiple Ways

The sample partitioning idea can be applied with more wallets than just MyMonero. Wallets with a nonstandard DSA are also more likely to have nonstandard fee calculation, `tx_extra` contents, and possibly other transaction uniformity defects. Through querying databases provided by neptune I have tentatively discovered some `tx_extra` formats associated with nonstandard DSAs.

Once labeled as such, those transactions can be excluded from the estimation because they contribute only noise. With the estimation approach I have outlined above, we can only recover the real spend age distribution from transactions that use a known decoy selection algorithm. If a wallet implementation is using an unknown non-standard DSA and distinguishes its transactions with other nonstandrad features, its transactions can be removed from the sample before estimation.

13.4 `wallet2` Integer Truncation Bug

Between Monero versions v0.14.1.0 and v0.17.2.2, `wallet2`'s DSA would integer-truncate a value that dictated the spread of the decoy distribution. This triggered a discontinuity in the distribution at the point in time that the value would be truncated to the next integer. The triggering is apparent at approximate block height 2,381,000 in

[8] https://github.com/mymonero/mymonero-core-cpp/pull/36
[9] https://www.getmonero.org/2021/09/20/post-mortem-of-decoy-selection-bugs.html

June 2021 in some of the plots of [Krawiec-Thayer et al., 2021]. I do not plan to investigate the trigger point since the reward is probably not worth the effort. If we wanted to investigate it, would could get an approximate number for the proportion of transaction constructed with `wallet2` around the time of the trucation trigger.

14 Future Work to Improve the Two-Step Estimator

A general note about estimators: If a single valid (i.e. consistent) estimator of a particular estimand exists, then multiple valid estimators exist almost always. In general what distinguishes the estimators may be the assumptions required for their validity and their variance and bias. My proposed two-step estimator is probably not the lowest-variance estimator of the estimand. The estimator "leaves information on the table" in a number of ways. In general, when more true information (do not impose false assumptions if avoidable!) about the structure of the statistical problem is considered by the estimator, its estimates become more precise. For example, when more true over-identifying restrictions are imposed by Generalized Method of Moments estimators, variance usually falls. I give three ways that the precision of the estimator can be improved by leveraging more information.

First of all, my proposed estimator does not use the fact that all rings in the same transaction almost certainly use the same decoy selection algorithm and therefore all belong to the same $F_k$component. (Note, however, that rings can be cached when a transaction is first constructed but not broadcast, freezing the ring in time.)[10] Every estimator that I have reviewed that attempts to estimate mixture models with repeated measurements nonparametrically, i.e. that attempts to estimate the model of Equation (4), has an equal number of observations for repeated measurements. If we could take into account the hierarchical transaction structure of the data, then the number of repeated measures $M + 1$ could vary according to the number of rings in a given transaction. As seen in **Section 12 Confidence Interval Estimation**, a larger $M + 1$ increases the precision of the Bonhomme-Jochmans-Robin estimator.

The second way is to convert the two-step estimator into a single-step joint estimator in which a single objective function is optimized to produce estimates in both steps simultaneously. Among maximum livelihood estimators, such an estimator is known as "Full Information Maximum Likelihood" (FIML) [Murphy & Topel, 1985]. Creating such an estimator would likely require a great deal of effort and skill in theoretical statistics.

The third way may or may not be mutually exclusive with the second. In the first step, we do not exploit the fact that we know quite a bit about several of the F_k components — specifically, that the component distributions have an admixture/contamination structure with a known parametric form of the largest sub-component (i.e. the DSA) for several of the components. There may be a way to include this information from the second step into the first step estimator.

There are a few recent papers on using Bayesian methods to model "mixture of mixtures" or "hierarchical mixture models"([Malsiner-Walli et al., 2017], [Argiento et al., 2020], [Frühwirth-Schnatter et al., 2021]).

I consider these possible improvements to the estimator to be out of scope for this initial OSPEAD CCS. The gain is not worth the effort at this time.

15 Rucknium Ratio Attack/MAP Decoder/Discriminant Analysis

An inebriated Homer Simpson once said, "To alcohol! The cause of, and solution to, all of life's problems." Similar to alcohol, an estimate of the real spend age distribution is the cause of, and solution to, many of Monero's statistical attack problems.

The potency of statistical attacks is a critical overall factor influencing the disclosure decision in **Section 17 Disclosure Considerations**. [Ronge et al., 2021] give the following metrics of statistical attack potency:

[10] https://libera.monerologs.net/monero-dev/20220622#c111774

Definition 3.1. *The guessing probability of* X *is*

$$\text{Guess}(\mathsf{X}) \coloneqq \max_X \Pr\left[\mathsf{X} = X\right].$$

This gives an upper bound on the probability than an (unbounded) adversary can "guess" the value of the random variable X correctly.

Definition 3.2. *The (average) conditional guessing probability of* X *given* Y *is defined as*

$$\text{Guess}(\mathsf{X}|\mathsf{Y}) \coloneqq \sum_Y \Pr\left[\mathsf{Y} = Y\right] \max_X \Pr\left[\mathsf{X} = X|\mathsf{Y} = Y\right].$$

This gives an upper bound on the probability that an (unbounded) adversary "guesses" the value of the random variable X correctly when given a sample of Y....

Theorem 4.4. *Let* $\mathcal{A}$ *be any computationally unbounded adversary, who inputs a ring* $\Pi(S)$ *(where* Π *[the ring sampler, i.e. Decoy Selection Algorithm,] is possibly subverted by* $\mathcal{A}$*) and some leakage* $\Lambda(S)$*, where* S *is sampled from the distribution* $\mathcal{S}$ *(possibly influenced or specified by* $\mathcal{A}$*), and outputs a guess* S'*. The probability of* $\mathcal{A}$ *correctly guessing the signer* S*, i.e.,* $S' = S$*, is upper bounded by*

$$\text{Guess}\left(\mathsf{S}|\Pi(\mathsf{S}), \Lambda(\mathsf{S})\right) = 2^{-\alpha(\mathsf{S},\Pi,\Lambda)}. \tag{11}$$

The right hand side of Equation (11) is not important for this discussion. $\Lambda(\mathsf{S})$, which represents side-channel information like view keys, logs of centralized exchanges, EAE/EABE attacks, etc., is out of scope. Therefore, we will deal with the simpler $\text{Guess}\left(\mathsf{S}|\Pi(\mathsf{S})\right)$. Possible active attacks on the DSA by adversary $\mathcal{A}$ like flooding/black marble attacks are also out of scope.

In my HackerOne submission I performed a Monte Carlo simulation to approximate the probability of success of the "Rucknium Ratio Attack" when the decoy selection algorithm diverges from the real spend age distribution, where I found $\text{E}\left[\widehat{\text{Guess}}_{RRA}\left(\mathsf{S}|\Pi(\mathsf{S})\right)\right] \approx 0.35$. The Rucknium Ratio Attack guesses that the most likely real spend is the ring member, compared to all other ring members in a ring, whose age x is at the maximum ratio $\frac{f_S(x)}{f_D(x)}$, i.e. the ratio of the real spend distribution to the decoy distribution. Unknown to me at the time, a few months prior [Aeeneh et al., 2021] had published an essentially identical attack they called the "Maximum A Posteriori (MAP) Decoder". They did not attempt to estimate the empirical potency of the attack because they did not think obtaining an estimate of the real spend age distribution $f_S(x)$ was possible. I will refer to this attack as the MAP Decoder since it is more descriptive of the technique and [Aeeneh et al., 2021] were first to publish. The Maximum A Posteriori Decoder is related to the Maximum A Posteriori (MAP) *estimator*. The MAP estimator find the statistical mode (i.e. the most frequent value) of the posterior distribution for a particular distribution [Bassett & Deride, 2019]. The MAP Decoder finds the most frequent value for the ratio of two distributions.

In their Equation 13, [Aeeneh et al., 2021] state the closed-form expression for the error probability of their attack. I will translate it into our notation. M is the number of decoys. Let $\boldsymbol{R} = \{r_1, r_2, \ldots, r_{M+1}\}$ be the set of ring members that have corresponding block height $\boldsymbol{B} = \{b_1, b_2, \ldots, b_{M+1}\}$. Without loss of generality, assume that the real spend is r_1. Let J be the number of blocks that have RingCT outputs. In other words, J is the maximum age of a possible ring member, in units of block height. Then when the real spend r_1 is at block height b_1, the probability that the MAP Decoder correctly guesses the real spend is:

$$\left(\sum_{j=1}^{J} f_D\left(j\right) \mathbf{1}\left\{\frac{f_S(j)}{f_D(j)} < \frac{f_S(b_1)}{f_D(b_1)}\right\}\right)^M \tag{12}$$

The $\mathbf{1}\left\{\frac{f_S(j)}{f_D(j)} < \frac{f_S(b_1)}{f_D(b_1)}\right\}$ indicator function term checks if a potential decoy drawn from the jth block would have a lower $\frac{f_S(x)}{f_D(x)}$ value than the real spend r_1. If the potential decoy does not have a lower value that the real spend,

then the real spend would not be guessed by the MAP Decoder. The $f_D(j)$ weights each $\mathbf{1}\left\{\frac{f_S(j)}{f_D(j)} < \frac{f_S(b_1)}{f_D(b_1)}\right\}$ term by the probability that the decoy selection algorithm would choose a decoy from that jth block height. Then the $\sum_{j=1}^{J}$ operator sums each weighted indicator function for every possible J blocks that a decoy could be chosen from. Then for the whole expression we need to calculate the joint probability for every decoy selected in the ring. We can take the Mth power of the expression for M decoys since the decoy selection algorithm (or at least that of `wallet2`) chooses decoys in an independent and identically distributed manner.

Expression 12 gives the probability that the MAP Decoder correctly guesses the real spend for a particular real spend r_1 at block height b_1. To calculate the expectation of $\text{Guess}_{MAPD}(\mathsf{S}|\Pi(\mathsf{S}))$ (i.e. the average) over all real spends, we weight Expression 12 by the probability that a user spends from each block height age $f_S(x)$ and then sum:

$$\mathrm{E}\left[\text{Guess}_{MAPD}(\mathsf{S}|\Pi(\mathsf{S}))\right] = \sum_{b_1=1}^{J} f_S(b_1) \left(\sum_{j=1}^{J} f_D(j)\, \mathbf{1}\left\{\frac{f_S(j)}{f_D(j)} < \frac{f_S(b_1)}{f_D(b_1)}\right\} \right)^M \tag{13}$$

Equation (13) apparently requires summation of J^2 elements. Given that J is in excess of one million, computation of (13) could be computationally expensive. However, by intelligently ordering the quantities to be compared, the computational expense can be reduced by several orders of magnitude. In `map-decoder-attack-potency.R` I perform the calculations with a naive implementation and an efficient one. With estimates of $f_S(x)$ and $f_D(x)$ from my HackerOne submission, Equation (13) gives $\mathrm{E}\left[\text{Guess}_{MAPD}(\mathsf{S}|\Pi(\mathsf{S}))\right] \approx 35.43693\%$, which is within about one hundredth of a percentage point of 35.4512%, the value I found from my original Monte Carlo simulation. When ring size is 16, $\mathrm{E}\left[\text{Guess}_{MAPD}(\mathsf{S}|\Pi(\mathsf{S}))\right] \approx 30.39\%$ with my HackerOne estimates of $f_S(x)$ and $f_D(x)$.

It is beneficial to have the Equation (13) closed-form expression of $\mathrm{E}\left[\text{Guess}_{MAPD}(\mathsf{S}|\Pi(\mathsf{S}))\right]$ with an efficient implementation rather than needing to resort to Monte Carlo simulation. Minimizing $\mathrm{E}\left[\text{Guess}_{MAPD}(\mathsf{S}|\Pi(\mathsf{S}))\right]$ is one of several criteria that I propose for determining the OSPEAD decoy selection algorithm later in **Section 21.6 Maximize Resistance to an Attack**. For a numerical minimization algorithm it is much better to have function evaluation computation time measured in seconds rather than minutes. There are also particular difficulties (though surmountable) with having the objective function be stochastic as would be the case with $\mathrm{E}\left[\text{Guess}_{MAPD}(\mathsf{S}|\Pi(\mathsf{S}))\right]$ estimated with Monte Carlo simulation.

My HackerOne estimate of $\mathrm{E}\left[\text{Guess}_{MAPD}(\mathsf{S}|\Pi(\mathsf{S}))\right]$ was based on the false simplifying assumption that the `wallet2` decoy selection algorithm was the only one being used on the Monero blockchain, using a simplified version of the Patra-Sen Inversion Estimator to obtain an estimate of $f_S(x)$. My proposed two-step estimator will produce a consistent estimate of $f_S(x)$ under the assumption that multiple decoy selection algorithms are being used. A more accurate estimate of $\mathrm{E}\left[\text{Guess}_{MAPD}(\mathsf{S}|\Pi(\mathsf{S}))\right]$ could be produced from this improved estimate of $f_S(x)$. I am not sure if this more accurate estimate of $\mathrm{E}\left[\text{Guess}_{MAPD}(\mathsf{S}|\Pi(\mathsf{S}))\right]$ would be greater than or less than the original HackerOne estimate. There are two factors that cut in opposite directions.

First, the original estimate would have conflated decoys from nonstandard decoy selection algorithms with real spends, potentially biasing the estimate of $\mathrm{E}\left[\text{Guess}_{MAPD}(\mathsf{S}|\Pi(\mathsf{S}))\right]$ upward. The magnitude of this source of bias is dependent on the divergence between the `wallet2` DSA and the nonstandard DSAs, plus the proportion of on-chain transactions that use the nonstandard DSAs.

On the other hand, the original estimate of $\mathrm{E}\left[\text{Guess}_{MAPD}(\mathsf{S}|\Pi(\mathsf{S}))\right]$ may have been biased downward because it did not take into account the variability of $f_S(x)$ over time. It gave an "average" over the time period under consideration. Assume for a moment that the `wallet2` DSA performed adequately for the average $f_s(x)$ over time. If $f_s(x)$ was highly variable from week to week, $\mathrm{E}\left[\text{Guess}_{MAPD}(\mathsf{S}|\Pi(\mathsf{S}))\right]$ could still be high since the MAP Decoder could exploit the extremes of each week.

We will not have an improved estimate of $\mathrm{E}\left[\text{Guess}_{MAPD}(\mathsf{S}|\Pi(\mathsf{S}))\right]$ until we have the improved estimate of $f_S(x)$.

15.1 Attack Optimality

[Aeeneh et al., 2021] claims in their Proposition 4 that the MAP Decoder attack is optimal in the sense that it minimizes the (guessing) error probability. If there there is no attack more powerful, then it may make sense to construct the OSPEAD DSA by minimizing the MAP Decoder effectiveness as I suggest in **Section 21.6 Maximize Resistance to an Attack**. It is not clear to me whether maximizing resistance to the most powerful attack also maximizes resistance to a less powerful attack. For example, the "guess newest heuristic" of [Kumar et al., 2017] and [Möser et al., 2018] is a less powerful attack that exploits *part* of the information about the discrepancy between $f_S(s)$ and $f_D(x)$.

In formal terms: Let $\Pi_{MAPD}(\mathsf{S})$ be the DSA that is constructed to minimize $\mathrm{E}\left[\mathrm{Guess}_{MAPD}\left(\mathsf{S}|\Pi(\mathsf{S})\right)\right]$ (with the constraint that $\Pi_{MAPD}(\mathsf{S})$ is static and parametric). Let $\Pi_{Alt}(\mathsf{S})$ be some alternative DSA that does not minimize $\mathrm{E}\left[\mathrm{Guess}_{MAPD}\left(\mathsf{S}|\Pi(\mathsf{S})\right)\right]$. Let $\mathrm{Guess}_{MAPD_{inferior}}$ be the guessing probability of an attack $MAPD_{inferior}$ that is inferior to the Map Decoder. Does there exist a $\Pi_{Alt}(\mathsf{S})$ and $MAPD_{inferior}$ such that

$$\mathrm{E}\left[\mathrm{Guess}_{MAPD}\left(\mathsf{S}|\Pi_{MAPD}(\mathsf{S})\right)\right] < \mathrm{E}\left[\mathrm{Guess}_{MAPD}\left(\mathsf{S}|\Pi_{Alt}(\mathsf{S})\right)\right]$$

(which is true by Proposition 4 of [Aeeneh et al., 2021]), yet also

$$\mathrm{E}\left[\mathrm{Guess}_{MAPD_{inferior}}\left(\mathsf{S}|\Pi_{MAPD}(\mathsf{S})\right)\right] > \mathrm{E}\left[\mathrm{Guess}_{MAPD_{inferior}}\left(\mathsf{S}|\Pi_{Alt}(\mathsf{S})\right)\right]?$$

If the answer is in the affirmative and an adversary can implement $MAPD_{inferior}$ but not $MAPD$, then it could be better to adopt something other than $\Pi_{MAPD}(\mathsf{S})$ as the new DSA. In such case, it might be better to use one of the other criteria I develop in **Section 21 Criteria for Best Fit**. That said, in the following discussion I lay out an argument that the MAP Decoder attack may have deep optimality properties.

My search of the statistics literature suggested that the MAP Decoder/Rucknium Ratio Attack is just a specific instance of a very old problem and class of solutions: Discriminant Analysis, a type of supervised classification ([McLachlan, 1992] [Fraley & Raftery, 2002]). As early as 1951, the year that [Fix & Hodges, 1989] was originally written under a U.S. Air Force contract, this problem was consider to be, "in a sense, completely solved." In 1989 [Fix & Hodges, 1989] was finally published in a journal due to its importance. After consistent estimation of the real spend age distribution, I take our problem as falling into the category of "subproblem (i)" below:

> The discrimination problem (two population case) may be defined as follows: a random variable Z, of observed value z, is distributed over some space (say, p-dimensional) either according to distribution F, or according to distribution G. The problem is to decide, on the basis of z, which of the two distributions Z has.
>
> The problem may be classified in various ways into subproblems. One pertinent method of classification is according to the amount of information assumed to be available about F and G. We may distinguish three stages:
>
> (i) F and G are completely known;
> (ii) F and G are known except for the values of one or more parameters;
> (iii) F and G are completely unknown, except possibly for assumption about the existence of densities, etc.
>
> Subproblem (i) has been, in a sense, completely solved. The solution is implicit in the Neyman-Pearson lemma (Neyman & Pearson, 1936), and was made explicit by Welch (1939). We may without loss of generality assume the existence of density functions, say f and g, corresponding to F and G, since F and G are absolutely continuous with respect to $F + G$. If f and g are known, the discrimination should depend only on $f(z)/g(z)$. An appropriate (positive) constant c is chosen, and the following rule is observed:

if $f(z)/g(z) > c$, we decide in favor of F;
if $f(z)/g(z) < c$, we decide in favor of G;
if $f(z)/g(z) = c$, the decision may be made in an arbitrary manner.

These procedures are known to have optimum properties with regard to control of probability of misclassification (probability of wrong decision). We shall refer to this as the 'likelihood ratio procedure', and denote it by $L(c)$....

The choice of c depends on considerations related to the relative importance of the two possible errors: saying Z is distributed according to G when in fact it is distributed to F, and conversely. Two choice of c have been widely advocated:

(a) take $c = 1$;
(b) choose c so that the two probabilities of error are equal.

Choice (a) has been called 'logical'; choice (b) yields the minimax procedure. In this paper we shall not concern ourselves with the choice of c, but shall assume that a given positive c is a datum of the problem.

"If f and g are known, the discrimination should depend only on $f(z)/g(z)$." This 70-year-old principle is used in the MAP Decoder attack. The main difference between MAP Decoder and this passage from [Fix & Hodges, 1989] is that the inequality criterion $f(z)/g(z) \gtreqless c$ is not governed by a constant c. In the MAP Decoder attack, c is variable, determined by the ratio $f(z)/g(z)$ for every other ring member in the ring, i.e. $c = \max_i\{f(z_i)/g(z_i)\}$.

[McLachlan et al., 2019] give a more general and modern explanation of the principle, placing it within the framework of finite mixture models:

The mixture model give by Equation (3) can be used to provide a model-based approach to clustering of the observed data in $\boldsymbol{y}_{obs}$ by conceptualizing that $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n$ come from a mixture in proportions $\pi_{1,}, \ldots, \pi_g$ of g groups $G_1, \ldots, G_g$ in which $\boldsymbol{Y}_j$ has density $f_1(\boldsymbol{y}_j; \boldsymbol{\theta}_1), \ldots, f_g(\boldsymbol{y}_j; \boldsymbol{\theta}_g)$, respectively. This is irrespective of whether these groups do externally exist.

For clustering purposes, each component in the mixture model given by Equation (3) is usually taken to correspond to a cluster. The posterior probability that the jth observation $\boldsymbol{y}_j$ arose from group G_i is given by the fitted posterior probability,

$$\tau_i(\boldsymbol{y}_j; \hat{\boldsymbol{\Psi}}) = \hat{\pi}_i f_i(\boldsymbol{y}_j; \hat{\boldsymbol{\theta}}_i)/f_i(\boldsymbol{y}_j; \hat{\boldsymbol{\Psi}}) \quad (i = 1, \ldots, g; j = 1, \ldots, n), \tag{14}$$

where $\hat{\boldsymbol{\Psi}}$ denotes an estimate of $\boldsymbol{\Psi}$.

A probabilistic clustering of the data $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n$ into g clusters can be obtained in terms of the fitted posterior probabilities of component membership $\tau_i(\boldsymbol{y}_j; \hat{\boldsymbol{\Psi}})(i = 1, \ldots, g)$.

An outright partitioning of the observations into g nonoverlapping clusters $C_1, \ldots, C_g$ is effectuated by assigning each $\boldsymbol{y}_j$ to the group G_i to which it has the highest estimated posterior probability of belonging. That is, the ith cluster C_i contains those observations $\boldsymbol{y}_j$ with $\hat{z}_{ij} = (\hat{\mathbf{z}}_j)_i = 1$, where

$$\begin{aligned} \hat{z}_{ij} &= 1, \qquad \text{if } i = \arg\max_h \hat{\tau}_h(\boldsymbol{y}_j; \hat{\boldsymbol{\Psi}}), \\ &= 0, \qquad \text{otherwise.} \end{aligned} \tag{15}$$

As the notation implies, $\hat{z}_{ij}$ can be viewed as an estimate of z_{ij} which, under the assumption that the observations come from a mixture of g groups $G_1, \ldots, G_g$, is defined to be one or zero according to whether $\boldsymbol{y}_j$ did or did not arise from $G_i(i = 1, \ldots, g)$.

The above rule for assigning the unclassified data points $\boldsymbol{y}_j$ to the g groups corresponds to the Bayes rule of allocation in the supervised classified case with known parameter vector $\boldsymbol{\Psi}$ ([McLachlan, 1992], section 1.3).

The $\hat{\pi}_i$ in equation (14) is the estimated mixing proportion of the ith density component. If we were to apply the assignment rule of (15) to a ring to classify ring members into the real spend and decoy distribution, the decoy distribution would always be heavily weighted since $\pi_D = \frac{M}{M+1}$, where M is the number of decoys. Thus, in many cases all ring members would be classified as decoys if this particular Bayes rule of allocation were to be used.

The general idea of this Bayes rule is packaged in a piece of advice given to physicians about diagnosis of disease: "When you hear hoofbeats in the night, look for horses — not zebras."[11] In other words, when you have some limited symptom data that is consistent with two diseases, i.e. $f_1(\boldsymbol{y}_j; \hat{\boldsymbol{\theta}}_1) = f_2(\boldsymbol{y}_j; \hat{\boldsymbol{\theta}}_2)$ for symptom data $\boldsymbol{y}_j$ and diseases 1 and 2, choose the diagnosis of the disease that is more prevalent in the population, i.e. choose the disease with the larger mixing proportion π_i.

Our problem is different in that we can use information from all $M+1$ observations in a set of repeated measures — all members of a ring — for classification. The general term for this setting is the "compound decision problem" [Copas, 1974]. To extend the metaphor, we know that there is exactly one zebra in the corral (there is exactly one real spend in a Monero ring). To re-imagine the MAP Decoder in the framework of the (15) assignment rule, we can instead make an assignment (a guess) based on:

$$\begin{aligned} \hat{z}_{ij} &= 1, \qquad \text{if } j = \arg\max_k \hat{\tau}_i(\boldsymbol{y}_k; \hat{\boldsymbol{\Psi}}), \\ &= 0, \qquad \text{otherwise.} \end{aligned}$$

In other words, apply the decision rule over the rows (ith dimension where i denotes the distribution component) of the $\hat{z}_{ij}$ matrix rather than over the columns (jth dimension where j denotes the observation). Thus, we consider each distribution (in our problem the two distributions: real and decoy) and then assign the most likely observation (ring member) to that distribution — we can ignore the decoy assignment since we are interested in the assignment of the real spend — rather than considering each observation and assigning the most likely distribution to it.

With a quick search I have not found a paper that covers our exact case of a compound decision problem in which the proportion of the sample belonging to each component is known exactly. Diving further into this corner of statistics may uncover more properties of the MAP Decoder such as how to maximize its error rate when one of the distributions, i.e. the decoy distribution, can be partially manipulated.

To conclude, it is likely that the MAP Decoder attack is the most dangerous statistical attack that exploits the timestamp information on the Monero blockchain. Therefore, I am personally leaning toward using the minimization of $\mathrm{E}\left[\mathrm{Guess}_{MAPD}\left(\mathsf{S}|\Pi(\mathsf{S})\right)\right]$ as the criteria to select the OSPEAD decoy selection algorithm.

16 Postestimation Classification of On-Chain Rings into DSAs

Given estimates of the mixture model components F_k, is it possible to probabilistically classify rings on the Monero blockchain according to their associated decoy selection algorithm? I believe it may be possible. There are two main reasons we might be interested in doing this.

The first is to better understand the degree of defects in transaction uniformity on the blockchain — the "anonymity puddles." Transaction uniformity defects themselves can be used as ammunition in statistical attacks against Monero user privacy. The second reason is to evaluate the safety of implementing an OSPEAD-derived DSA in a new release of the Monero `wallet2` software at a time other than a hard fork. It could be possible to probabilistically distinguish between transactions constructed with the new DSA and the old DSA, which could endanger user privacy. Without a hard fork, users who use the GUI or CLI wallet and wallet developers who use `wallet2` to construct transactions are not required to upgrade to the new version. The question is whether the cure would be worse than the disease:

[11] https://quoteinvestigator.com/2017/11/26/zebras/

$$\Pr\left(MAPD_{wallet2\_DSA}\right) \overset{?}{\lesseqgtr} \Pr\left(MAPD_{OSPEAD\_DSA} \cup Classifier_{wallet2\_DSA/OSPEAD\_DSA}\right) \tag{16}$$

where $MAPD_{wallet2\_DSA}$ is the event that the real spend is guessed by the MAP Decoder attack with the existing DSA, $MAPD_{OSPEAD\_DSA}$ is the same with the OSPEAD-derived DSA, and $Classifier_{wallet2\_DSA/OSPEAD\_DSA}$ is the event that a DSA-based classifier correctly guesses the real spend by distinguishing between transactions using the existing DSA and the OSPEAD-derived DSA. Further data analysis is required because the direction of the inequality in (16) must be resolved empirically rather than theoretically.

Inequality (16) is not very formal. It does not consider the "adoption curve" over time of users updating to a new `wallet2` version. A more complete decision criteria would consider the risk over time and an educated guess about the adoption curve. An educated guess could be based on an estimate of the rate of adoption of the bug fixes to the decoy selection algorithm written by jberman in 2021, using the Bonhomme-Jochmans-Robin estimator.

There are a few candidate techniques for probabilistic classification of rings using their repeated measures structure. [Bagui et al., 2006] describe the state of the literature as of 2006:

> Various nonparametric procedures are available in situations where independent repeated measurements are available on the entity to be classified; they have been considered by Das Gupta [7], Hudimoto [15], Kanazawa [18], Gupta and Kim [13], Lin [17], Govindarajulu and Gupta [12], and Haack and Govindarajulu [14] among others. DasGupta [7] proposed an allocation rule using the Wilcoxon statistic. Govindarajulu and Gupta [12] considered an allocation rule based on linear rank statistics for the s-group problem....
>
> Bagui [2] considered the problem of classification of m independent multiple observations under the mixture sampling scheme which is common in prospective studies and diagnostic situations. Bagui et al. [3] proposed the allocation of multiple observations using a sub-sample approach under mixture sampling. Bagui and Mehra [4] used a rank nearest neighbor (RNN) classification rule to classify univariate independent multiple observations between two classes. The present article deals with the classification of m independent (multivariate) multiple observations under a separate (or conditional) sampling scheme.

I consider exploration of ring classification to be out of scope of this OSPEAD CCS. It would require much additional research. Future work could cover this ground.

17 Disclosure Considerations

Given the potential threat to retroactive user privacy of an estimate of the real spend age distribution combined with the MAP Decoder attack, there is a question of whether to disclose the estimator. A sophisticated adversary who pays close attention likely would already be aware of the MAP Decoder attack since it was published in [Aeeneh et al., 2021]. All that is missing is an estimate of the real spend age distribution. The decoy selection algorithm of course would be plainly visible in Monero's source code, so it is a matter of disclosing how the parameters of the algorithm was chosen.

The process to decide on disclosure is not very formal at this point. It is not clear what exact body of individuals plays the role of deciding to disclose or not. After I completed my initial investigations in August/September 2021 I submitted my findings through HackerOne in compliance with Monero's Vulnerability Response Process. After discussion, the VRP manager deferred to my judgment about whether to publicly disclose immediately. I chose not to disclose because the risk to user privacy was still not completely understood and we had no fix ready. It would make sense for the revoew panel to give input on about the advisability of disclosure.

As I stated in Section 15, we will not have an improved estimate of the potency of the MAP Decoder attack until we have the improved estimate of the real spend distribution Therefore, we do not need to make a final call at this point in time. Below I discuss some pros and cons of disclosure.

17.1 Pros

17.1.1 Greater peer review of OSPEAD

Disclosure would enable wider peer review of my proposed solution. Disclosure of this document and my HackerOne submission would potentially put more eyes on the OSPEAD estimator. Furthermore, if we choose disclosure I am willing to prepare a manuscript for submission to a privacy-focused journal for a traditional peer review process. We could then benefit from suggested improvements to OSPEAD.

17.1.2 More attention to this research area

Many papers such as [Egger et al., 2022][Ronge et al., 2021], and [Yu et al., 2019] have suggested a "partitioning" (i.e. a single bin) DSA, partially since they did not think it was feasible to estimate the real spend age distribution. Disclosure of OSPEAD could revitalize research on mimicking DSAs.

Nonparametric mixture models are an active area of statistical research. With disclosure, we might be able to catch the interest of theoretical statisticians to help us refine and strengthen the OSPEAD approach.

An estimate of the real spend age distribution may also be useful to potential future research on churning.

17.1.3 User trust

Community trust in the security of the code and the privacy of its protocol is essential. Many community members might not accept concealment of any part of the process of how certain designed decisions were determined. When I submitted my CCS proposal, my suggestion that parts of OSPEAD be remain a secret was very controversial.[12]

Community members' concern is not without some basis in reality. In the past, malicious parties have used "suggested" cryptographic parameters to effectively create backdoors, such as the suspected NSA backdoor in Dual_EC_DRBG.[13]

On the other hand, there is precedent for publication of certain security parameters without explicit justification because the justification would reveal attacks, such as with the discovery of differential cryptanalysis:[14]

> The discovery of differential cryptanalysis is generally attributed to Eli Biham and Adi Shamir in the late 1980s, who published a number of attacks against various block ciphers and hash functions, including a theoretical weakness in the Data Encryption Standard (DES). It was noted by Biham and Shamir that DES was surprisingly resistant to differential cryptanalysis but small modifications to the algorithm would make it much more susceptible.
>
> In 1994, a member of the original IBM DES team, Don Coppersmith, published a paper stating that differential cryptanalysis was known to IBM as early as 1974, and that defending against differential cryptanalysis had been a design goal. According to author Steven Levy, IBM had discovered differential cryptanalysis on its own, and the NSA was apparently well aware of the technique. IBM kept some secrets, as Coppersmith explains: "After discussions with NSA, it was decided that disclosure of the design considerations would reveal the technique of differential cryptanalysis, a powerful technique that could be used against many ciphers. This in turn would weaken the competitive advantage the United States enjoyed over other countries in the field of cryptography."

17.1.4 Probabilistic attacks may be irrelevant for certain threat models

For many user threat models, probabilistic/statistical attacks based on the divergence of the real spend age distribution from the decoy selection algorithm may not be relevant. Plausible deniability is still likely intact even with statistical attacks. Due to Monero's anonymous nature, we cannot really know the threat models that users confront, but the existence of statistical attacks might not be a threat to most users' privacy.

[12] https://www.reddit.com/r/Monero/comments/py8ub3/ccs_proposal_ospead_fortifying_monero_against/
[13] https://en.wikipedia.org/wiki/Dual_EC_DRBG
[14] https://en.wikipedia.org/wiki/Differential_cryptanalysis

[Deuber et al., 2022] attempt to formalize the reliability of certain classes of de-anonymization attacks and heuristic analysis of cryptocurrency transactions, including Bitcoin, Zcash, and Monero. They rate different techniques by relevance to law enforcement: "A high relevance means that the attack is very likely actually employed by law enforcement; medium means that it is not known whether the attack is used but we see potential use-cases; low means that we do not see a potential use-case because the exploited weakness has been fixed or the attack would require too much effort." For example, they rate the "multi-output assumption" against Monero, which is statistical, as having "medium" relevance. They do not assess the relevance of decoy timing attacks since they consider the guess newest attack rectified with the implementation of the gamma distribution.

17.1.5 Monero adversaries may already be using these techniques

It is possible that chain analysis companies already have a good estimate of the real spend age distribution and are aware of the MAP Decoder attack. In such case, there would be little to gain from withholding the information. In 2020, the U.S. Internal Revenue Service awarded a contract to Chainalysis and Integra to develop Monero and BTC Lightning Network tracing tools.[15] Ciphertrace claims to have Monero tracing tools. [16] On the other hand, an Elliptic representative claimed that no chain analysis companies can trace Monero.[17]

17.1.6 An improved DSA mimics the real spend age distribution

If an adversary is aware of the MAP Decoder attack and is observing public discussion of the OSPEAD research, then they may assume that the new proposed public DSA closely mimics the real spend age distribution, which could enable the MAP Decoder attack. The main ambiguity that an adversary would deal with would be the question of how closely the new DSA mimics the real spend age distribution, the time period over which the distribution was fit, and how stable the distribution is over time. In other words, an adversary would not necessarily know how well a MAP Decoder attack would perform on past and future time periods of blockchain data.

17.2 Cons

17.2.1 Retrospective analysis

A principal challenge of cryptocurrency privacy, compared to privacy in other types of protocols, is that historical data exists forever publicly. Therefore, passive retroactive analysis is always a potential threat, regardless of whether any particular techniques existed at the time of the target transactions.

In a recent case, Swedish-Russian national Roman Sterlingov was arrested in the Los Angeles airport in 2021 on charges of running a BTC mixing service.[18] A significant part of the evidence supporting the arrest is based on chain analysis of BTC transactions that occurred in 2011.

17.2.2 Probabilistic attacks may be relevant for extreme threat models

Certain users may be in environments where probabilistic analysis does not rise to the standard of proof for using legal investigative tools, filing charges, and/or conviction in a court of law. On the other hand, in other environments potential adversaries of Monero users may not feel themselves constrained by the rule of law. One-party states, dictatorships, absolute monarchies, and criminal gangs come to mind. Probabilistic analysis also opens the door to false positives, false accusations, and mis-targeted robberies. Even users in jurisdictions that ostensibly respect the

[15] https://news.bitcoin.com/chainalysis-and-integra-win-1-25-million-irs-contract-to-break-monero/
[16] https://ciphertrace.com/heads-up-monero-v15-hard-fork-update/
[17] https://monero.observer/elliptic-liat-shetret-no-blockchain-analytics-company-can-track-monero/
[18] https://www.wired.com/story/bitcoin-fog-roman-sterlingov-blockchain-analysis/

rule of law may be falsely targeted. There is a long history of the use of unreliable pseudoscience by law enforcement in such jurisdictions [Lilienfeld & Landfield, 2008].

17.2.3 The potency of probability-weighted transaction graph attacks is unknown

The unpublished MRL Research Bulletin #0011 raised the possibility of overlaying a set of probability weights on the edges of the Monero transaction graph to strengthen graph-based attacks on anonymity. I consider evaluating the risk of graph-based attacks to be out of scope of OSPEAD at this time. I personally do not have a strong grasp of graph theory.

Graph-based attacks may present a different risk from the "single hop" risk that my overall analysis focuses on. As such, it could present a blind spot whose danger we may fully realize only in the future. A few recent papers on non-probablistic graph attacks make me less nervous about this blind spot, however. [Ye et al., 2020] executed the chain reaction attack of [Kumar et al., 2017] on newer data. See their Section 3.2.1. They found that the share of transactions that were fully or partially traceable using this attack fell to almost zero when RingCT was introduced in January 2017. [Vijayakumaran, 2021] performed the Dulmage-Mendelsohn (DM) decomposition on the Monero transaction graph, which "is optimal in the sense that it eliminates every output-key image association which is incompatible with the transaction history." [Vijayakumaran, 2021] found "For pre-RingCT outputs in Monero, the DM decomposition technique performs better than existing techniques. For RingCT outputs in Monero, the DM decomposition technique has the same performance as existing techniques, with only five out of approximately 29 million outputs being identified as spent."

[Egger et al., 2022] approach the question from a theoretical perspective. They attempt to put a bound on the necessary size of rings to defeat certain graph attacks when the decoy selection algorithm is a partitioning one rather than a mimicking one. I discussed my interpretation of the paper in the July 6, 2022 MRL meeting.[19] They argued that to defeat global graph-based attacks in a transaction graph with $|U|$ transaction outputs (and a partitioning DSA), the ring size should be at least $\ln(2 \cdot |U|) + \sqrt{2 \ln(2 \cdot |U|)}$. The current $|U|$ of Monero of 75 million would require ring size 25 in this framework. From the preceding set of empirical and theoretical results we may tentatively hope that adding probability weights to graph attacks might not significantly harm Monero user privacy. Note that isthmus asserts "anyways, unfortunately Egger et al's optimistic takeaway does not translate to the heuristics framework described in the doc, since NRL's work uses transaction metadata to establish priors for partitioning the graph by fingerprint, whereas the partitioning samplers in their work are oblivious to that information".[20]

18 OSPEAD Requests to Monero C++ Developer(s)

Several tasks that would greatly help OSPEAD can only be completed by someone who can read and write C++ code, which I cannot. mj-xmr is willing and able to complete these tasks. The tasks are here in decreasing order of priority:

1. Converting the `wallet2` decoy selection algorithm into a closed-form probability density function, i.e. $f_{wallet2,D}$. OSPEAD cannot be done at all without this step. mj-xmr and I have made progress on this task already.[21]

2. Write a fast C++ implementation of the Bonhomme-Jochmans-Robin estimator. With the current MATLAB/Octave implementation, bootstrapping and Monte Carlo simulations are not computationally feasible. Therefore, we cannot really know the precision of the two-step estimator without a faster implementation. In

[19] https://libera.monerologs.net/monero-research-lab/20220706#c117351
[20] https://libera.monerologs.net/monero-research-lab/20220817#c137450
[21] https://github.com/mj-xmr/monero-mrl-mj/tree/master/decoy

Octave testing with a single CPU core, with $M = 10$ decoys, $N = 1000$ rings, $K = 3$ distinct decoy selection algorithms, and $I = 10$ evaluation points, the estimation completed in about 24 hours. Assuming that the estimation time scales linearly with N, for a single week of data a single CPU core would take about a year of wall clock time to complete the estimation.

3. Determine a method to fingerprint MyMonero transactions by their fees. According to jberman, "simplest/quickest way would be to run their fee algorithm at various times to get a range of what fees their txs would be. hardest way would be to definitively calculate the plausible fees it could have chosen for each tx. they can likely be pinpointed with effort"[22] As I stated in **Section 13.2 MyMonero Fee Fingerprinting**, fingerprinted MyMonero transactions could help with the estimations.

4. Converting the MyMonero decoy selection algorithm into a closed-form probability density function, i.e. $f_{MyMonero,D}$. This is necessary for OSPEAD if we want to have a decoy selection algorithm that combines the real spend age distribution of `wallet2` users and MyMonero users as in Equation (7). It would also help overcome the "label swapping" problem discussed in **Section 12.2 Variance of the Second Step Estimator**.

5. A fast implementation of a forecast evaluator and simulator. mj-xmr has developed `tsqsim` in C++, which compares very favorably to existing methods coded in slower languages, but certain modifications are needed such as dealing with multivariate inputs. [23]

[22] https://libera.monerologs.net/monero-research-lab/20220615#c109279
[23] https://www.reddit.com/r/Monero/comments/tl5w1b/tsqsims_benchmark_new_tool_designed_for_monero/

Part II

Disclosed Portion

19 Plain English Explanation of the Problem

19.1 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 3: 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 3 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.[24] 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].

19.2 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 2: Adequate Decoy Selection Algorithm

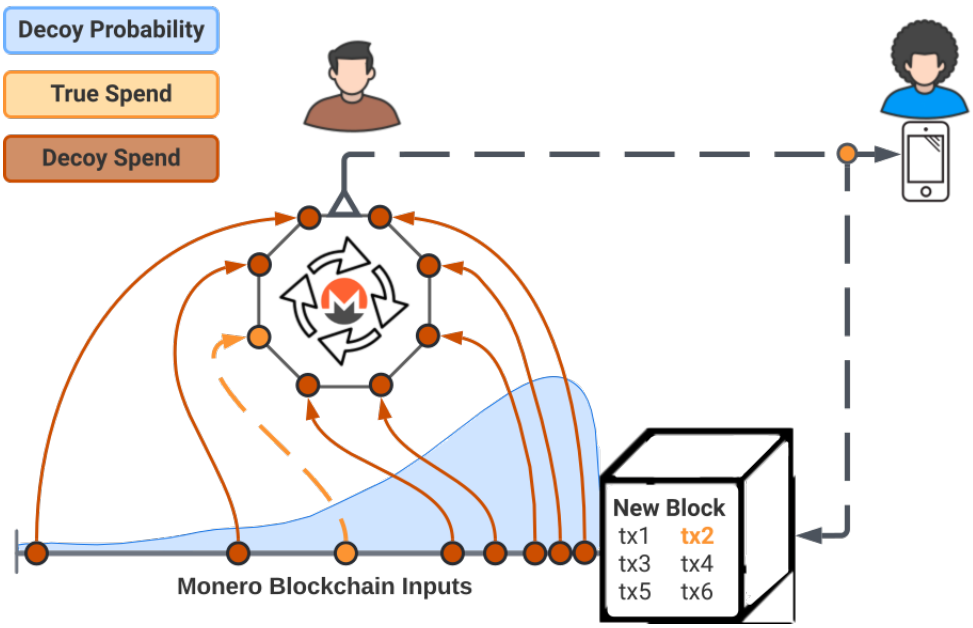


Image designed by ACK-J (https://github.com/ack-j)

The height of the blue shape in Figure 2 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.

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

Figure 3 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 3: Defective Decoy Selection Algorithm

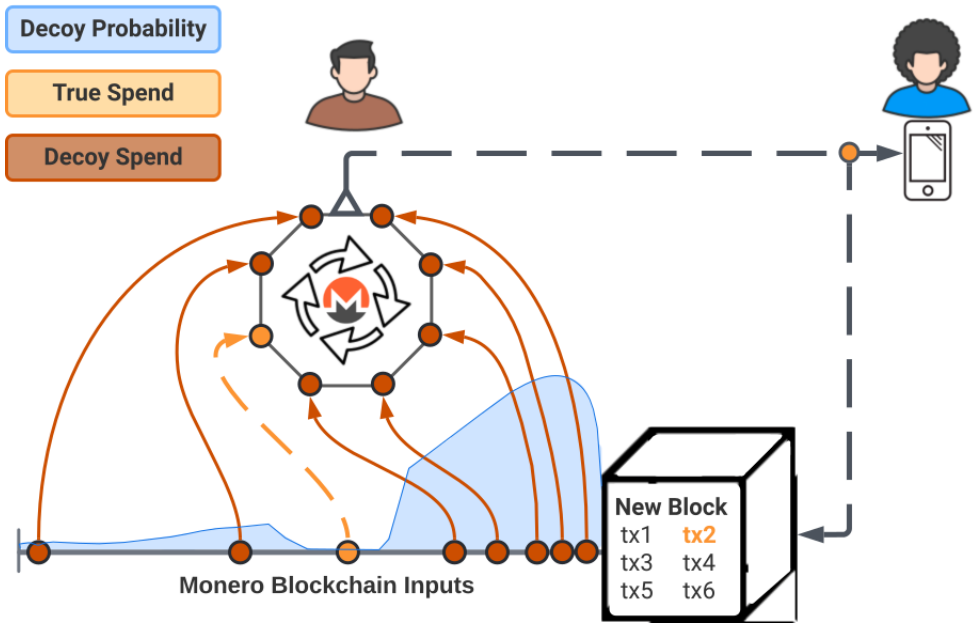


Image designed by ACK-J (https://github.com/ack-j)

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.

20 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.

21 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.[25] 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:

[25] 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

$$L(\theta, a) = \begin{cases} (a - \theta)^2 & \text{if } a < \theta \\ 10(a - \theta)^2 & \text{if } a \geq \theta. \end{cases} \tag{17}$$

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

$$R(\theta, \delta) = \mathrm{E}_\theta L(\theta, \delta(\mathbf{X})). \tag{18}$$

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 (18). 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 (17) 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 (18), [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 24 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.[26] 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:

$$w(\theta_i, \lambda) = \lambda\theta_i + (1 - \lambda)\tfrac{1}{N} \tag{19}$$

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.

21.1 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.

[26] 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:

$$L_{FGT_\alpha}(\boldsymbol{\theta}, \boldsymbol{a}, \lambda) = \frac{1}{N} \sum_{i=1}^{N} \mathbf{1}\left\{a_i < \theta_i\right\} w(\theta_i, \lambda) \left(\frac{\theta_i - a_i}{\theta_i}\right)^{\alpha} \tag{20}$$

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.

21.2 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:

$$u_{CRRA_\eta}(y) = \begin{cases} \frac{y^{1-\eta}}{1-\eta} & \eta \geq 0, \eta \neq 1 \\ \ln(y) & \eta = 1 \end{cases} \tag{21}$$

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:

$$L_{Welfare_\eta}(\boldsymbol{\theta}, \boldsymbol{a}, \lambda) = (-1)\,\frac{1}{N}\sum_{i=1}^{N} w(\theta_i, \lambda)\left(u_{CRRA_\eta}\left(\min\left\{\frac{a_i}{\theta_i}, 1\right\}\right)\right) \tag{22}$$

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 $\boldsymbol{\eta} = \{0.5, 1, 2, 5\}$.

21.3 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:

$$L_{Gini}(\boldsymbol{\theta}, \boldsymbol{a}, \lambda) = \frac{\sum_{i=1}^{n}\sum_{j=i+1}^{n} w(\theta_i, \lambda) \cdot w(\theta_j, a_j, \lambda) \cdot \left|\min\left\{\frac{a_i}{\theta_i}, 1\right\} - \min\left\{\frac{a_j}{\theta_j}, 1\right\}\right|}{\sum_{i=1}^{n} w(\theta_i, \lambda) \sum_{i=1}^{n} w(\theta_i, \lambda) \cdot \min\left\{\frac{a_i}{\theta_i}, 1\right\}} \tag{23}$$

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

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 $\frac{a_i}{\theta_i}$.

21.4 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:

$$\text{Ge}_{min} = \frac{\frac{1}{2}m(m+1)}{\frac{r_{max}}{r_{min}} + m} \tag{24}$$

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

$$\text{Ge}_{min\ (optimizer)} = \frac{r_{max}}{r_{min}} \tag{25}$$

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 (25) will also maximize (24)

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

$$L_{Worst\,case}(\boldsymbol{\theta}, \boldsymbol{a}) = \frac{\max\limits_{i\in\{1,\ldots,N\}} \left(\frac{\theta_i}{a_i}\right)}{\min\limits_{i\in\{1,\ldots,N\}} \left(\frac{\theta_i}{a_i}\right)} \tag{26}$$

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.

21.5 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:

$$L_{MLE}\left(\boldsymbol{\beta}|\mathbf{x}\right) = (-1) \cdot \sum_{i=1}^{n} \log g\left(x_i|\boldsymbol{\beta}\right) \tag{27}$$

21.6 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

$$L_{Attack}\left(f_D(x)\right) = \mathscr{P}\left(f_D(x)\right) \tag{28}$$

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.[27] 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 $f\left(x|\boldsymbol{\beta}\right)$ under consideration.

21.7 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:

$$
\begin{array}{ll}
\forall L \in \mathcal{L}, & \text{Loss functions} \\
\forall f(x|\boldsymbol{\beta}) \in \mathcal{F}, & \text{Parametric distributions} \\
\forall \lambda \in \boldsymbol{\lambda}, & \text{Weight parameters for } w(\theta_i, \lambda) \\
\forall \alpha \in \boldsymbol{\alpha}, & \text{Exponents for FGT poverty indicator} \\
\forall \eta \in \boldsymbol{\eta}, & \text{Parameters for CRRA utility} \\
 & \\
\min\limits_{\boldsymbol{\beta}\in\mathcal{B}} L\left(f\left(x|\boldsymbol{\beta}\right)\right) & \text{Numerical optimization problems} \\
 & \\
\text{where} & \\
\mathcal{L} = \{L_{FGT_\alpha}, L_{Welfare_\eta}, L_{Gini}, L_{Worst\,case}, L_{MLE}, L_{Attack}\} & \text{Equations (20), (22), (23), (26), (27), (28)} \\
\mathcal{F} = \{\text{Roughly 40 PDF/PMFs that have support } [0,\infty)\} & \\
\boldsymbol{\lambda} = \{0, 0.5, 0.9, 0.95, 0.99, 0.999, 0.9999, 1\} & \\
\boldsymbol{\alpha} = \{1, 2, 3\} & \\
\boldsymbol{\eta} = \{0.5, 1, 5, 10, 50\} & \\
\mathcal{B} = \{\text{Sets of parameters associated with each element of } \mathcal{F}\} &
\end{array}
\tag{29}
$$

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 $f\left(x|\boldsymbol{\beta}\right)$ 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 (29), is about 3,000.[28]

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.

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

[28] 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, $|L_{FGT_\alpha}(\boldsymbol{\theta}, \boldsymbol{a}, \lambda)| = |\boldsymbol{\lambda}| \cdot |\boldsymbol{\alpha}| = 24$, $\left|L_{Welfare_\eta}(\boldsymbol{\theta}, \boldsymbol{a}, \lambda)\right| = |\boldsymbol{\lambda}| \cdot |\boldsymbol{\eta}| = 40$, $|L_{Gini}(\boldsymbol{\theta}, \boldsymbol{a}, \lambda)| = |\boldsymbol{\lambda}| = 8$, $|L_{Worst\,case}(\boldsymbol{\theta}, \boldsymbol{a})| = 1$, $|L_{MLE}\left(\boldsymbol{\beta}|\mathbf{x}\right)| = 1$, and $|L_{RRA}\left(f_D(x)\right)| = 1$. Therefore, $|\mathcal{L}| = 75$ and hence $|\mathcal{L}| \cdot |\mathcal{F}| = 3,000$.

22 Dry Run with Old M oser et al. (2018) Data

To see how the minimizers in (29) 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 (29). 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 4: Performance of Dry Run with Old M oser 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 4 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 5 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 5 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 M oser 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 5: Optimized Parameter Values of Dry Run with Old M oser 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 4: Optimized $f_D(x)$ for loss function L_{FGT_α}, $\alpha = 1$

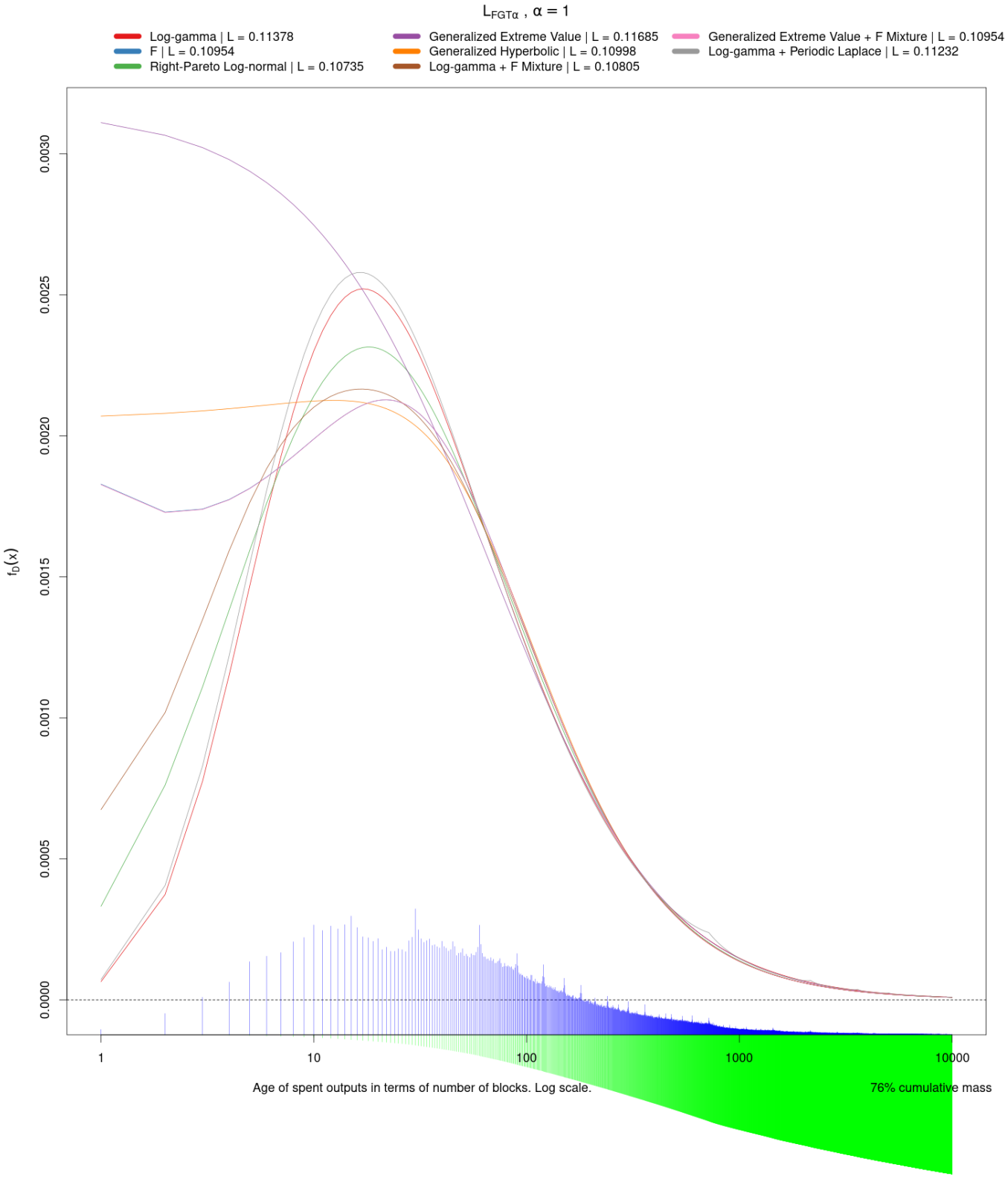


Figure 5: Optimized $f_D(x)$ for loss function L_{FGT_α}, $\alpha = 2$

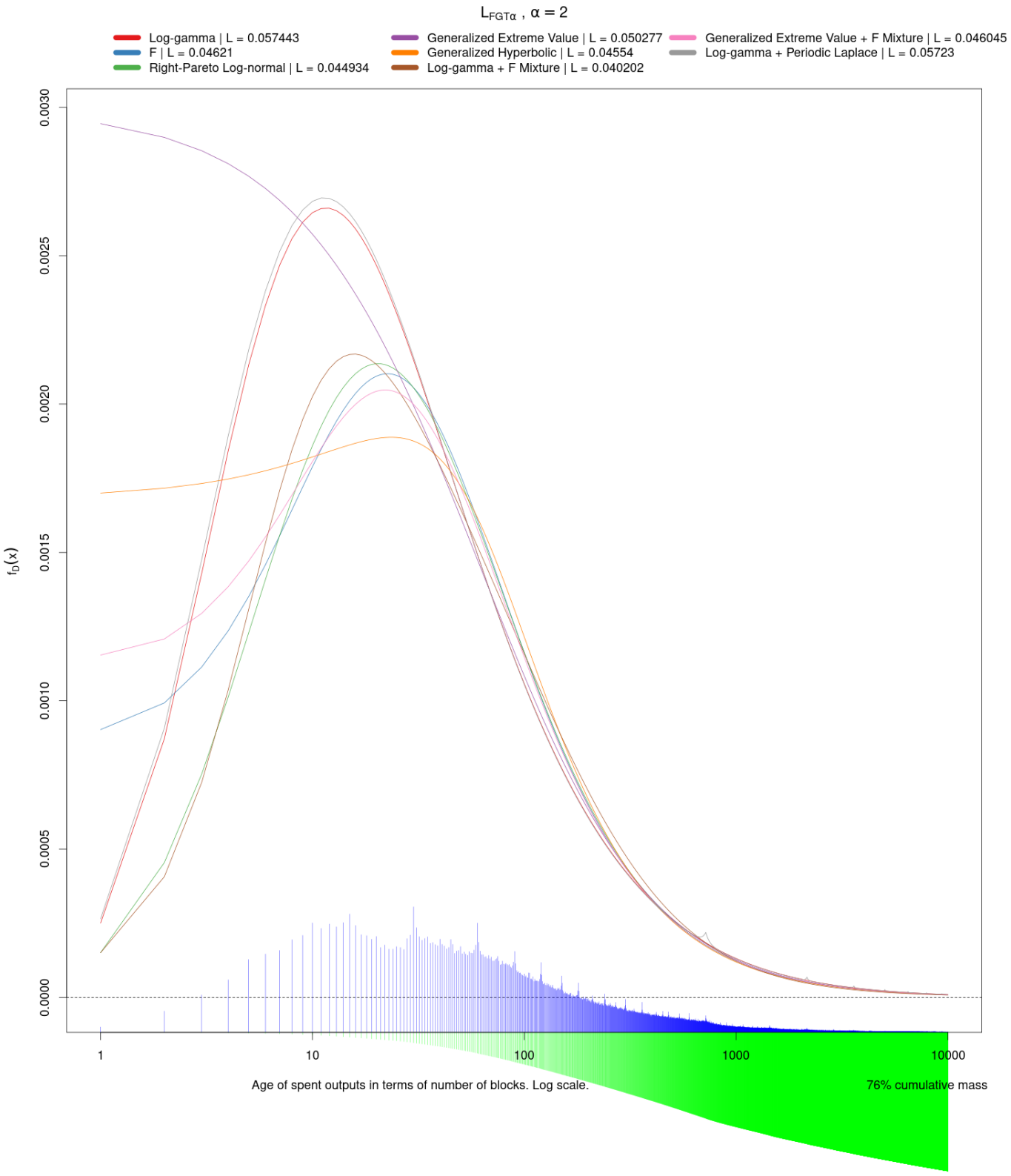


Figure 6: Optimized $f_D(x)$ for loss function $L_{Welfare_\eta}$, $\eta = 0.5$

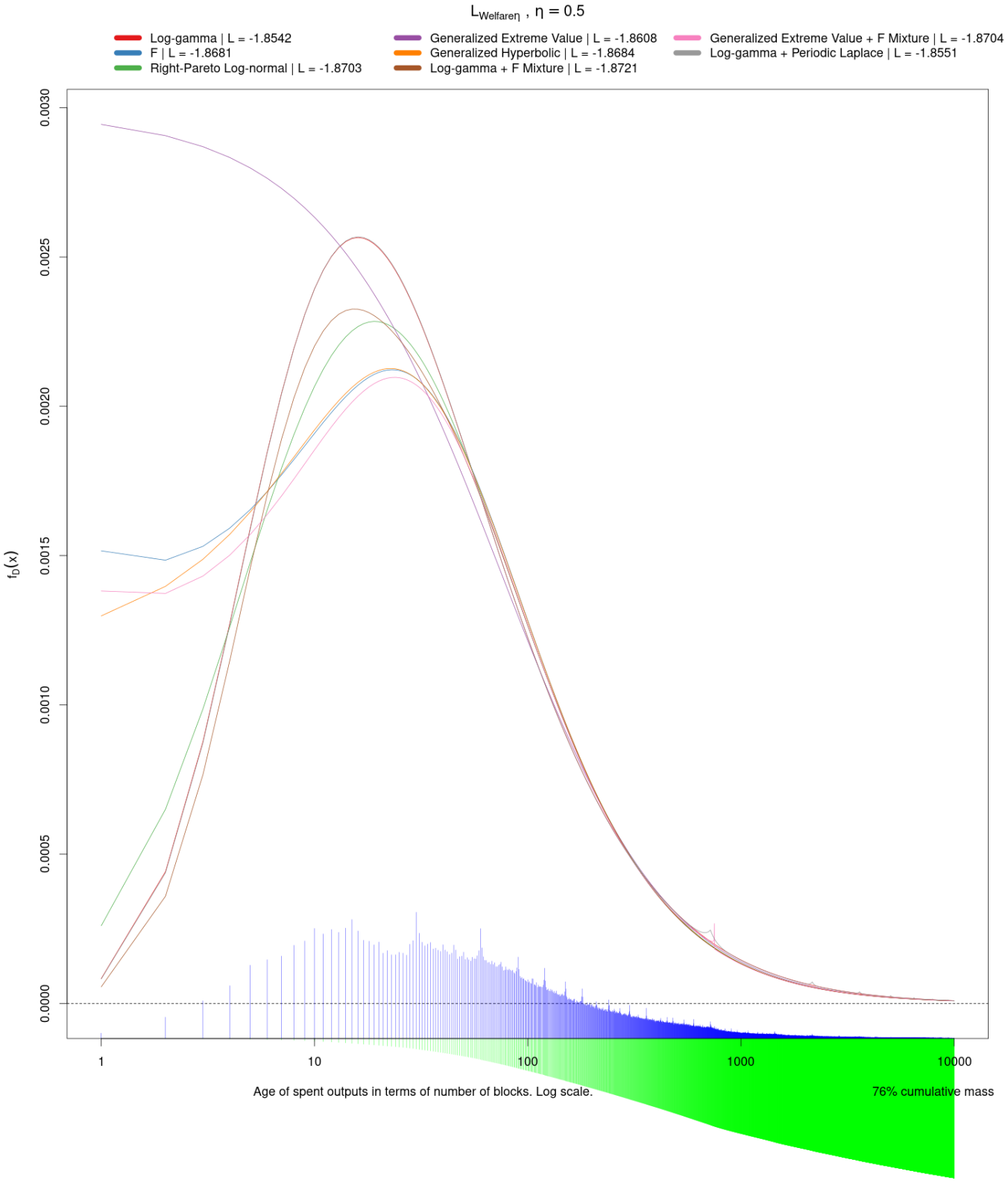


Figure 7: Optimized $f_D(x)$ for loss function $L_{Welfare_\eta}$, $\eta = 1$

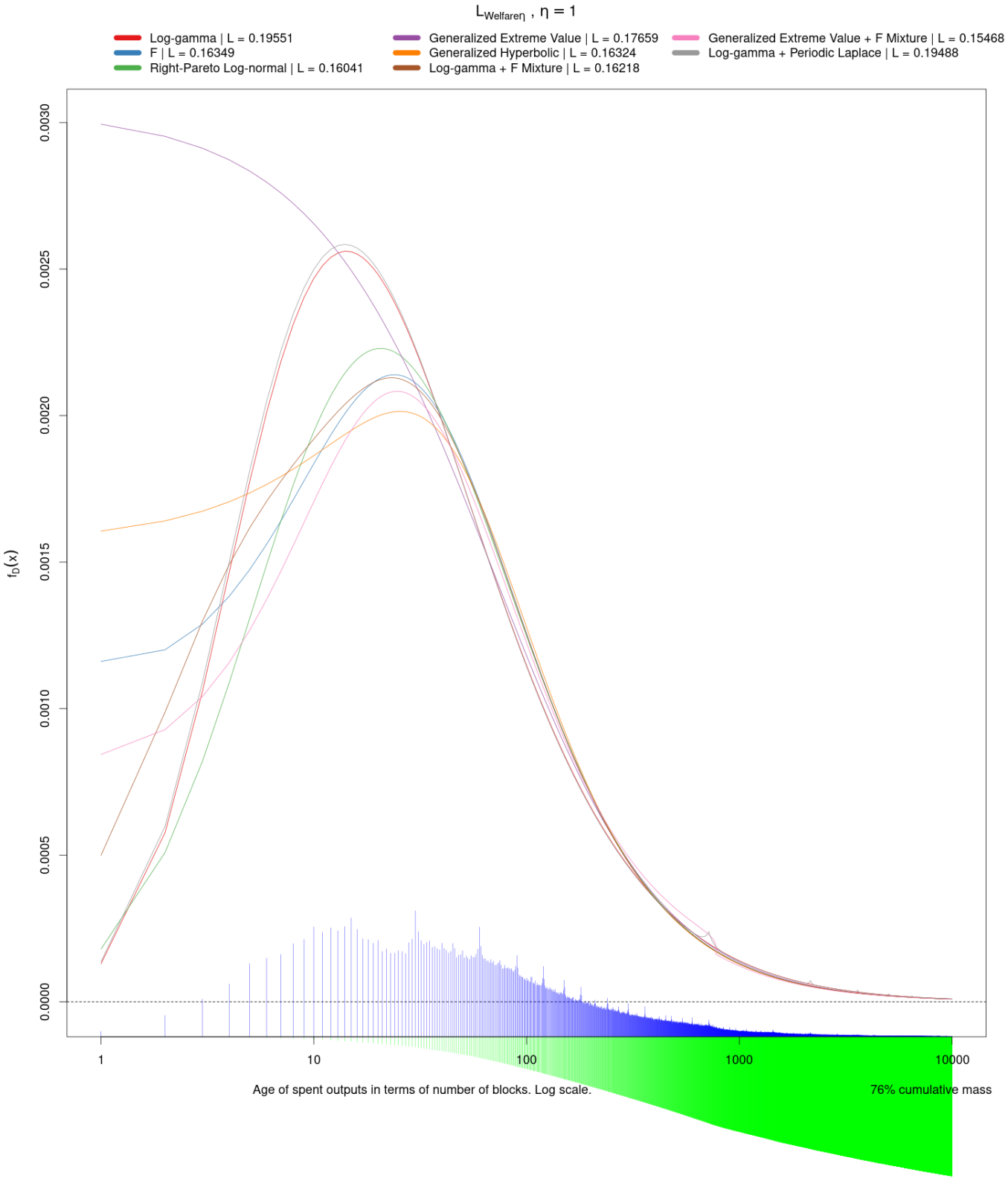


Figure 8: Optimized $f_D(x)$ for loss function L_{MLE}

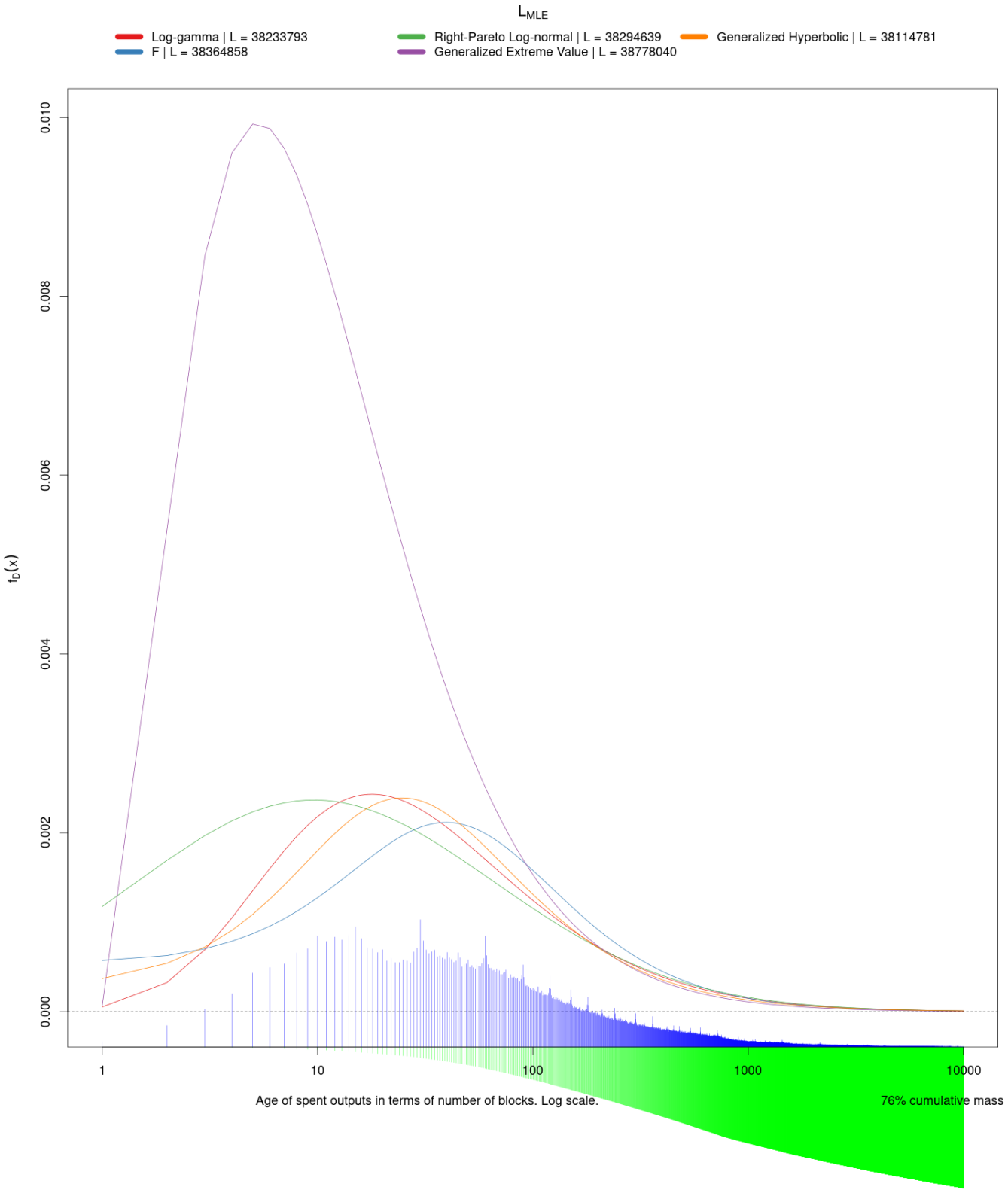


Figure 9: Optimized $f_D(x)/f_S(x)$ for loss function L_{FGT_α}, $\alpha = 1$

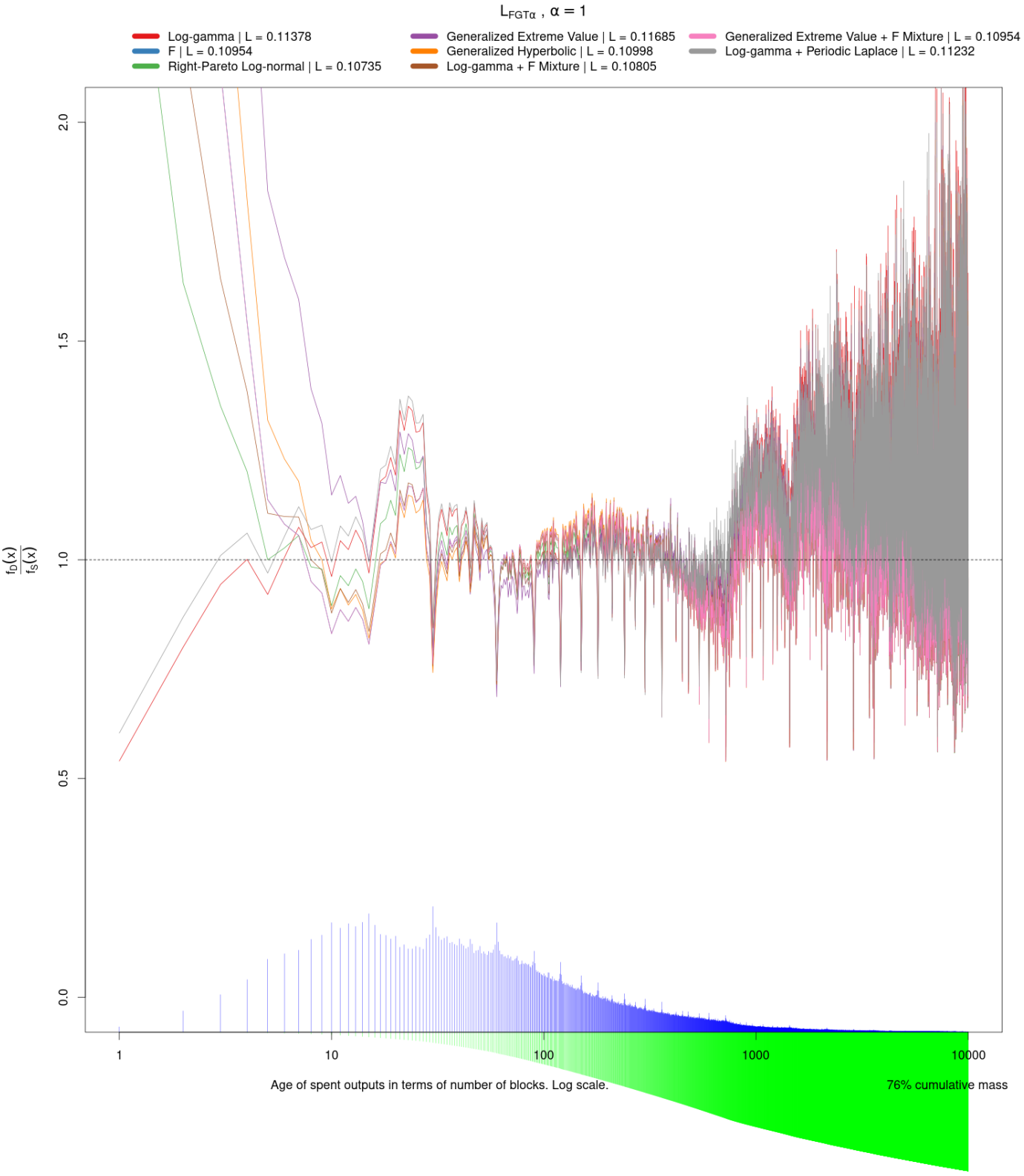


Figure 10: Optimized $f_D(x)/f_S(x)$ for loss function L_{FGT_α}, $\alpha = 2$

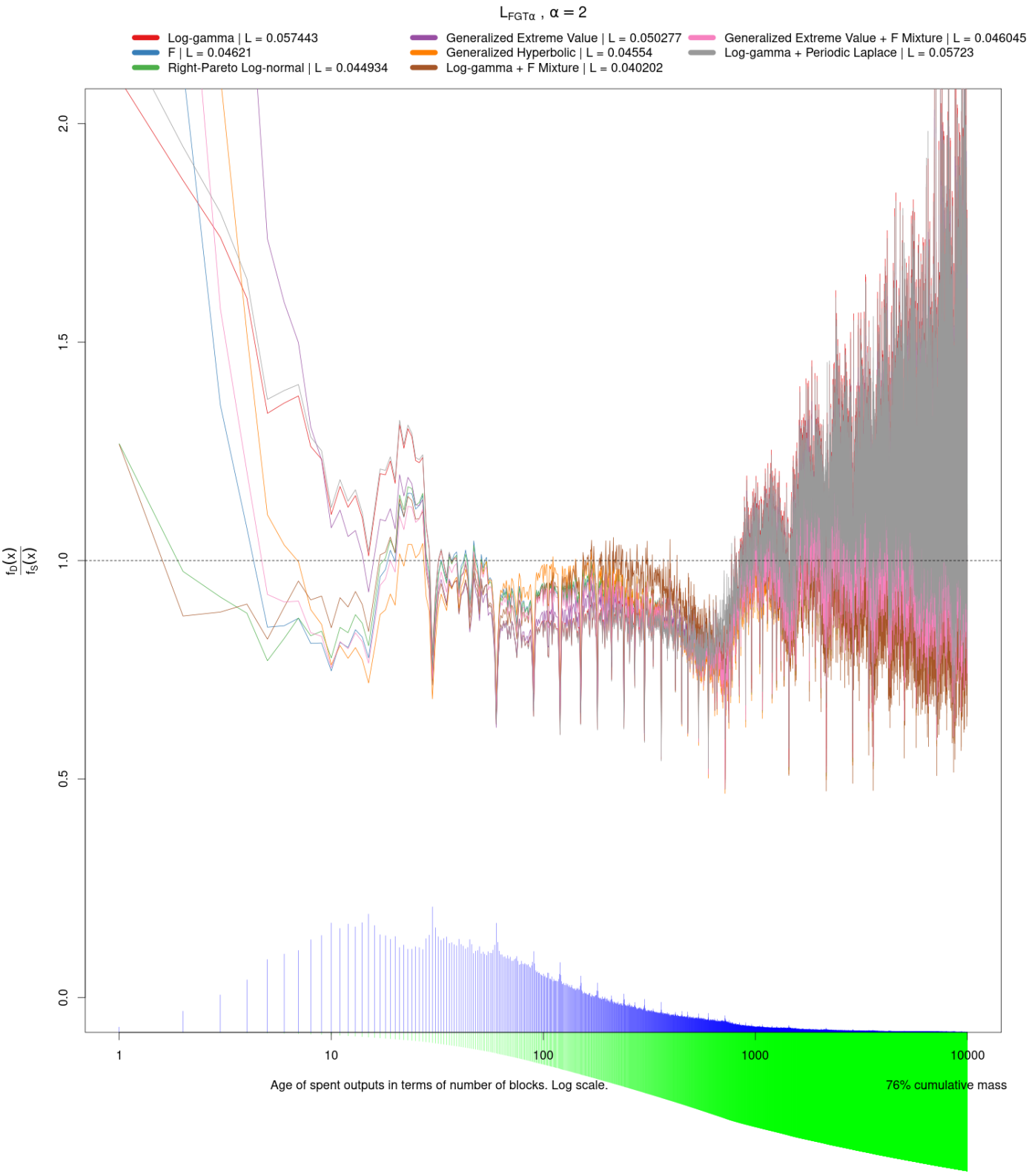


Figure 11: Optimized $f_D(x)/f_S(x)$ for loss function $L_{Welfare_\eta}$, $\eta = 0.5$

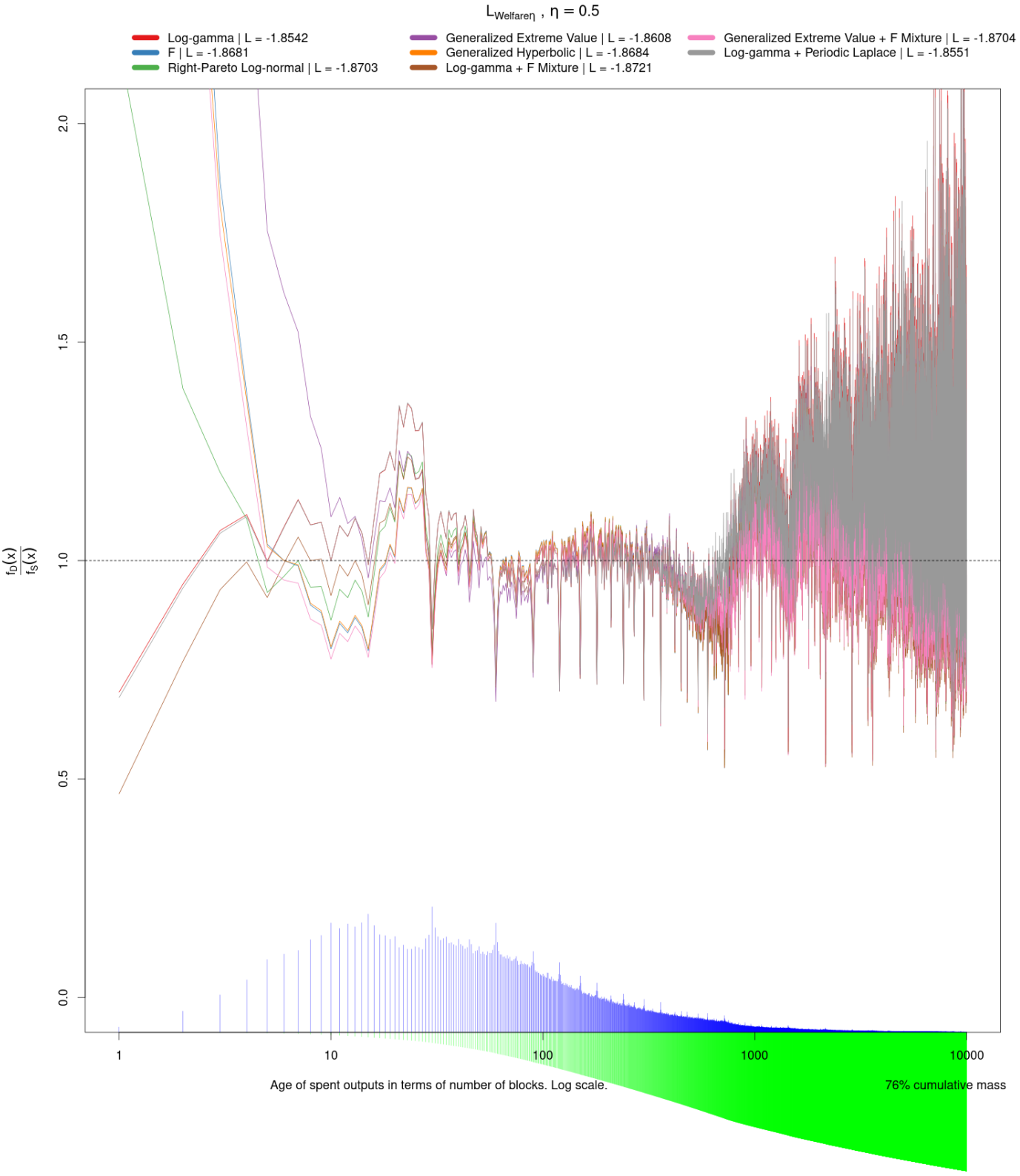


Figure 12: Optimized $f_D(x)/f_S(x)$ for loss function $L_{Welfare_\eta}$, $\eta = 1$

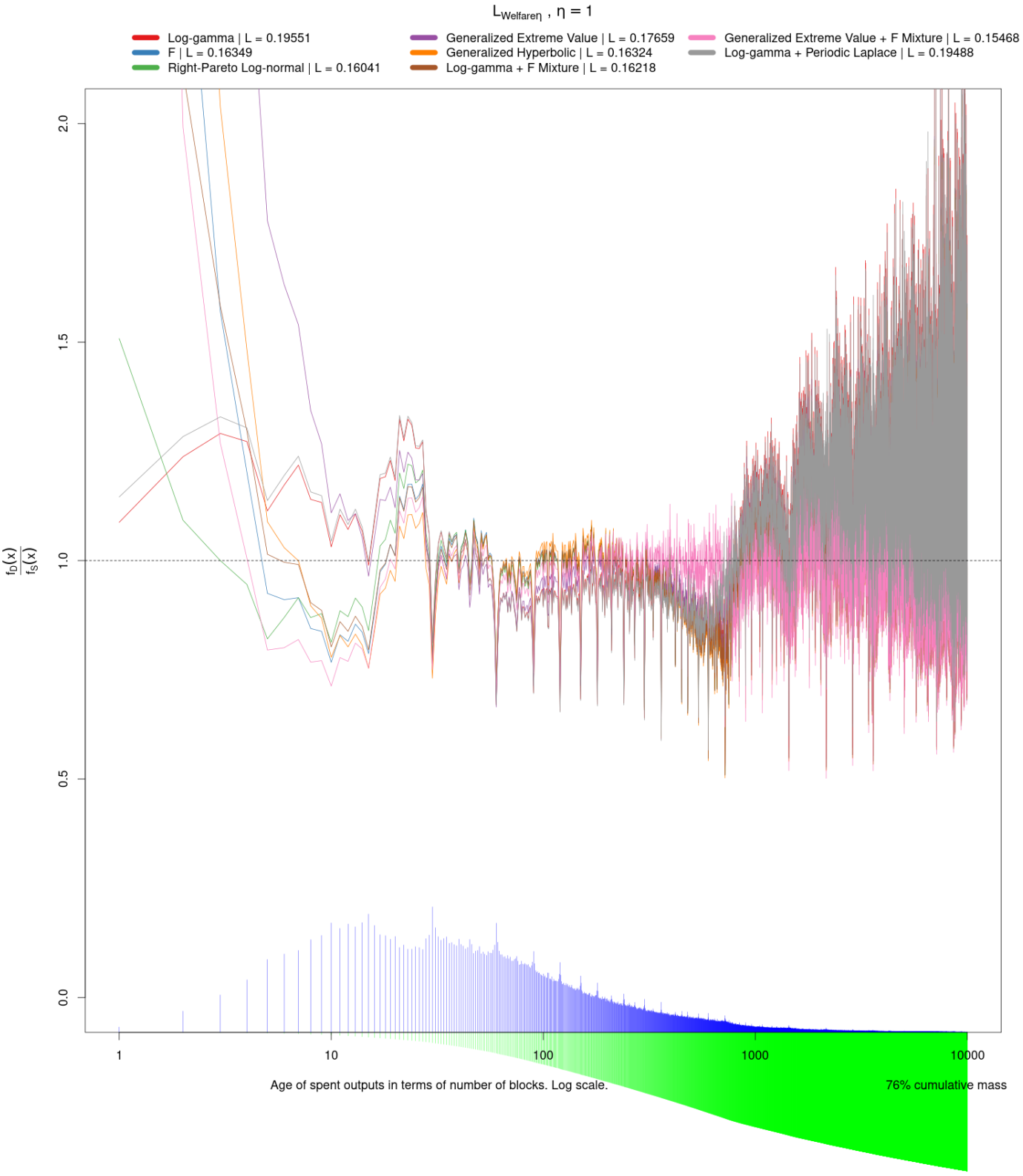
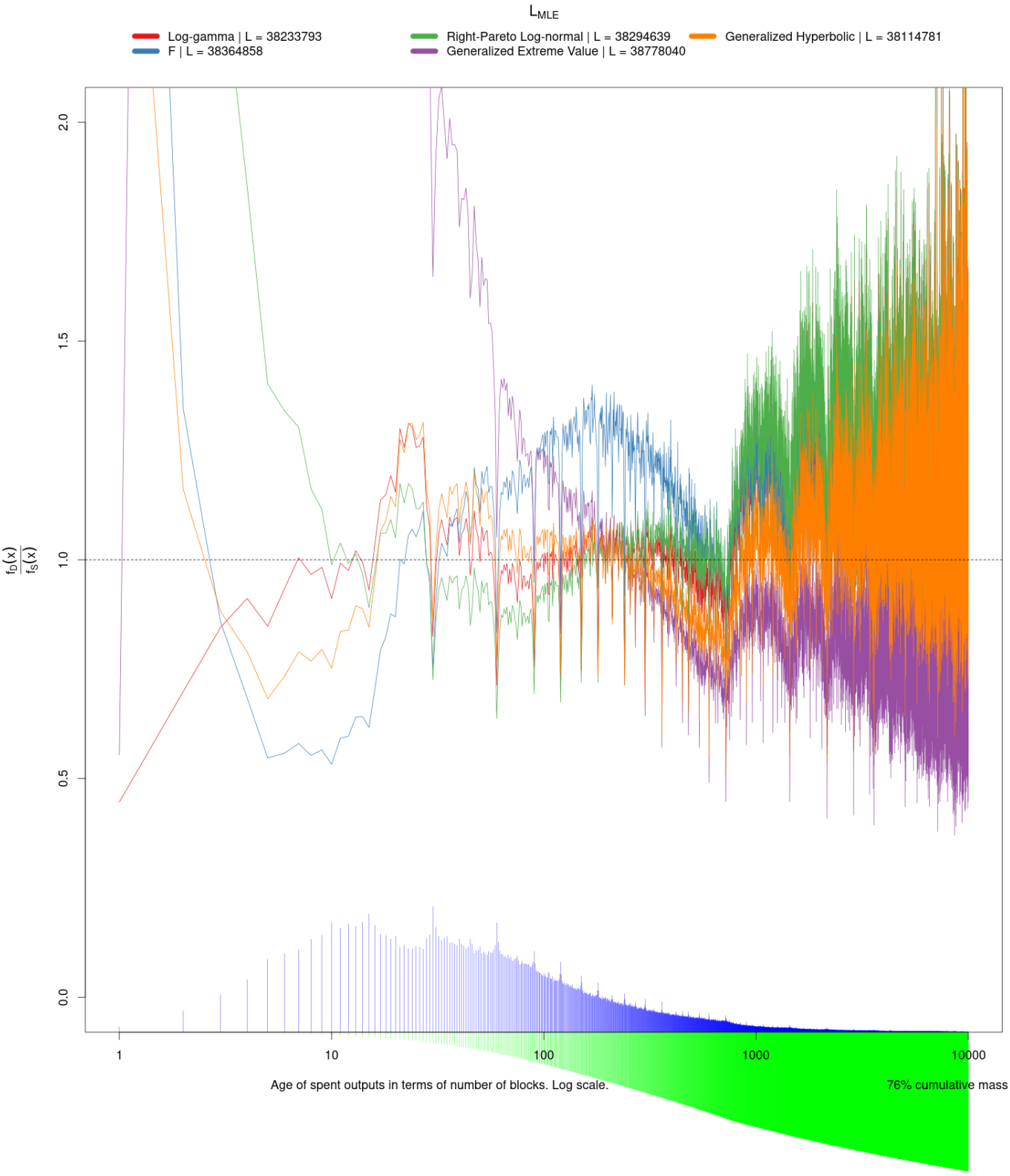


Figure 13: Optimized $f_D(x)/f_S(x)$ for loss function L_{MLE}



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

In **Section 21 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.

24 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.[29] 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 (18) from **Section 21 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

[29] 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.[30] 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.

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

Much like in **Section 22 Dry Run with Old M oser 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.

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

UTXO-based cryptocurrency usage as payment by percent (payment processors and merchants)							
	Service	Bitpay[31]	CoinCards[32]	Bitrefill[33]	Cake Pay[34]	Travala[35]	Shodan[36]
	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.[37]

25.1 Summary Characterization of Distributions Over Time

Figures 15 to 18 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.[38]

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:

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

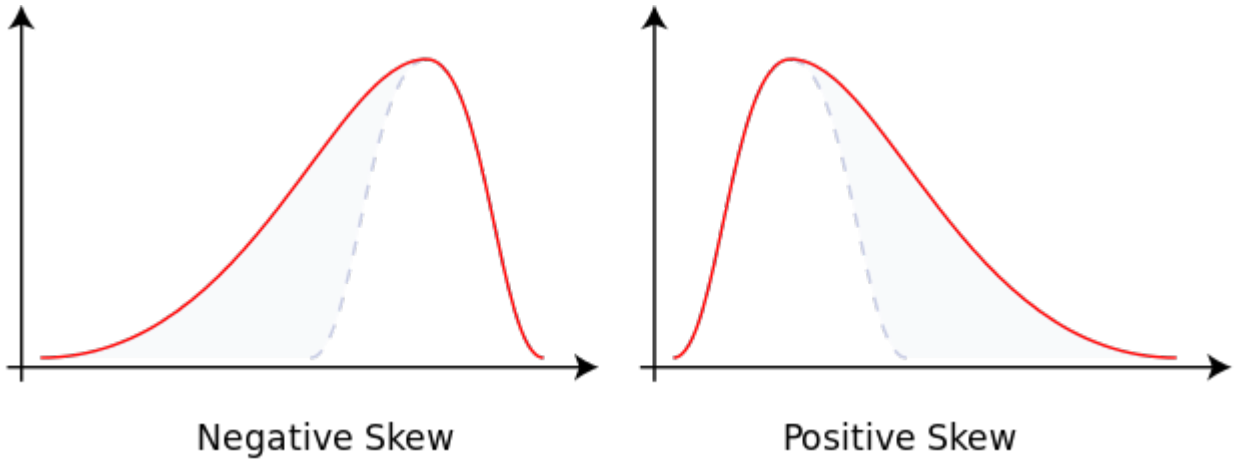
The skewness is the third standardized moment (standardized by the standard deviation σ):

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

The kurtosis is the fourth standardized moment:

[31] https://bitpay.com/stats/
[32] https://twitter.com/CoinCards/status/1555286172385116164
[33] https://youtu.be/bkjEcSmZKfc?t=549
[34] https://twitter.com/cakewallet/status/1565370179906838528
[35] https://travala-dashboard.com/
[36] https://blog.shodan.io/accepting-crypto-a-vendor-perspective/
[37] https://ccs.getmonero.org/proposals/gingeropolous_zenith_storage.html
[38] https://en.wikipedia.org/wiki/ISO_week_date

Figure 14: 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 15: BTC Summary Characterization of Spend Age Distribution Over Time

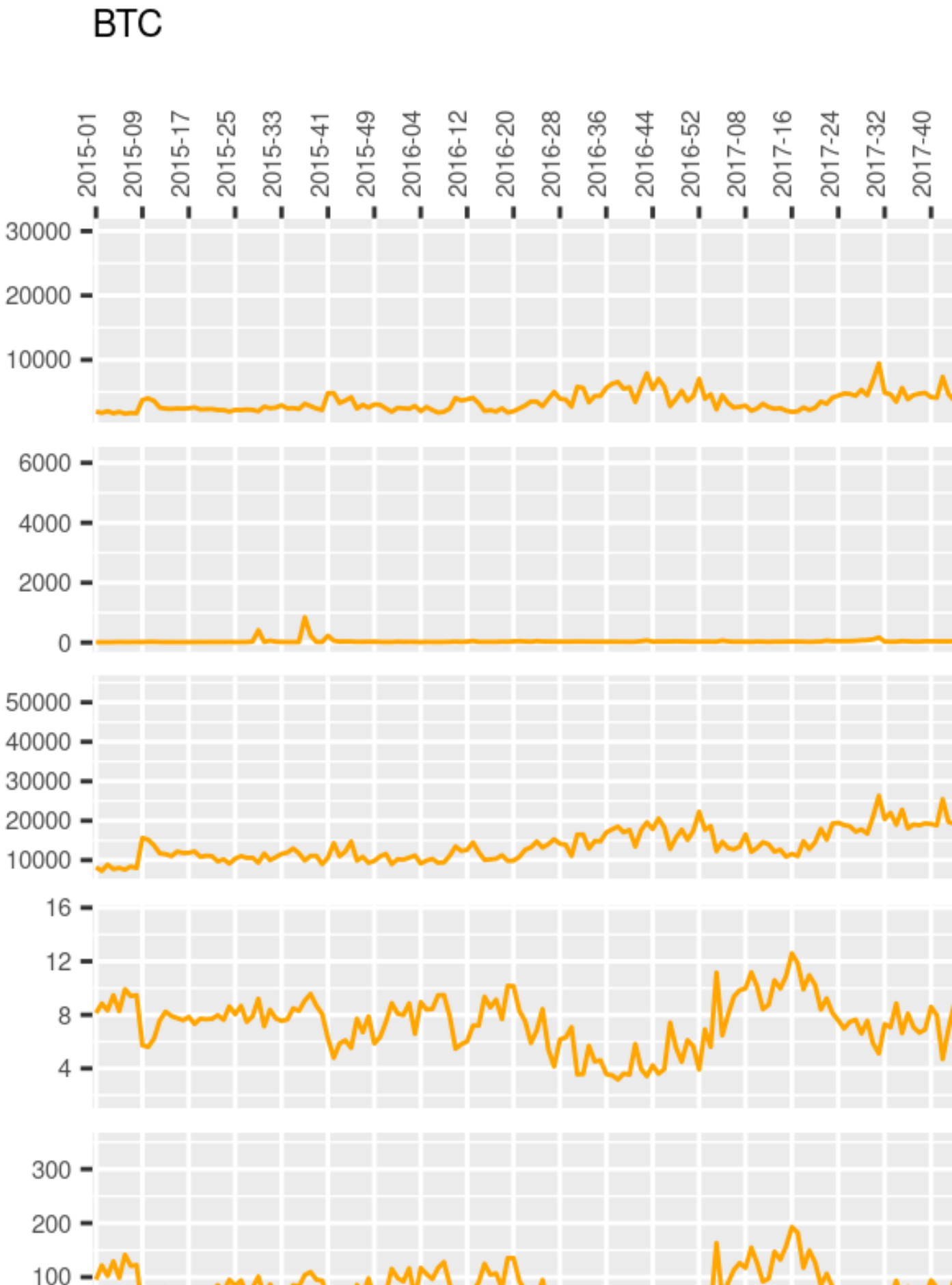


Figure 16: BCH Summary Characterization of Spend Age Distribution Over Time

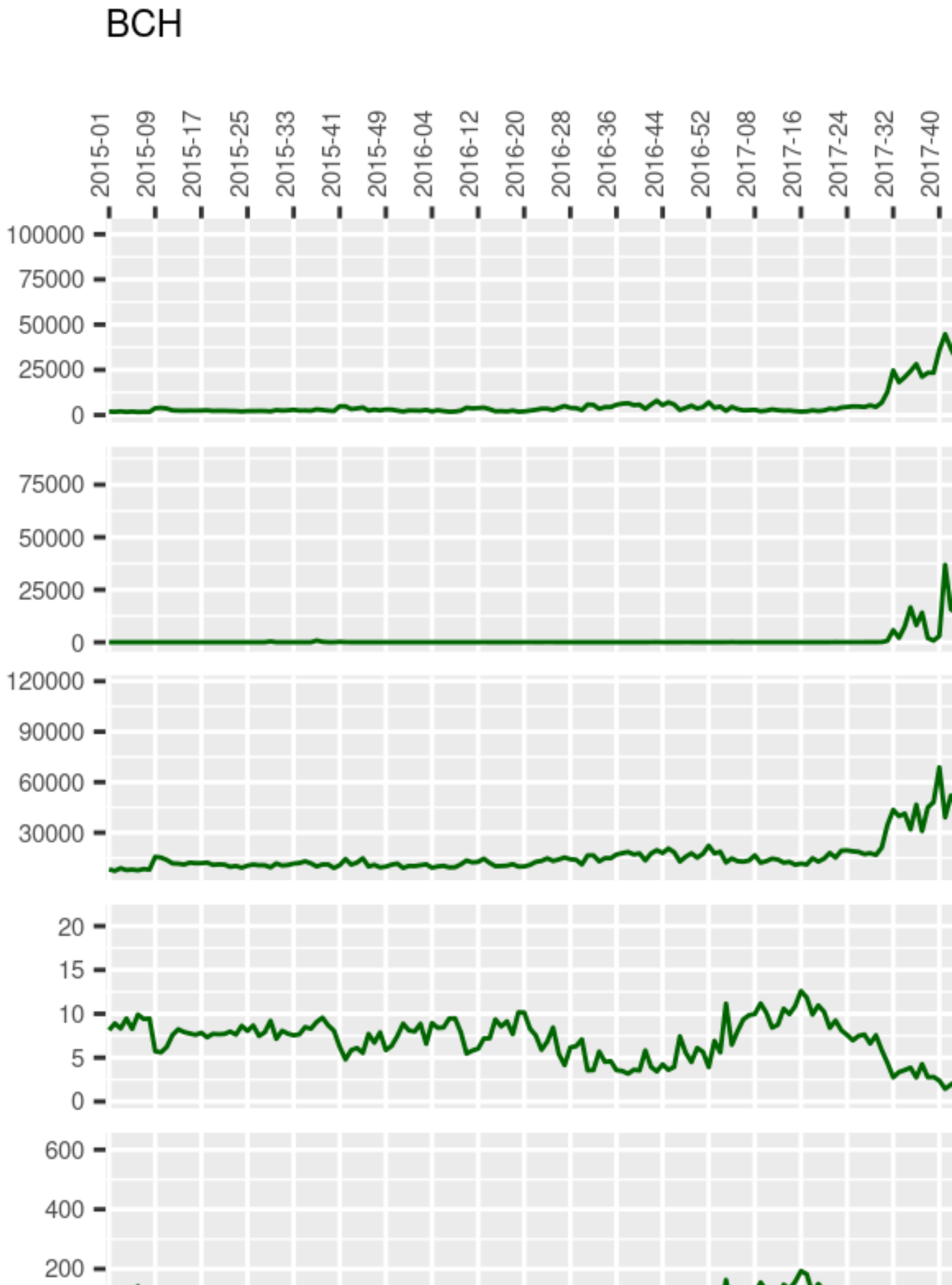


Figure 17: LTC Summary Characterization of Spend Age Distribution Over Time

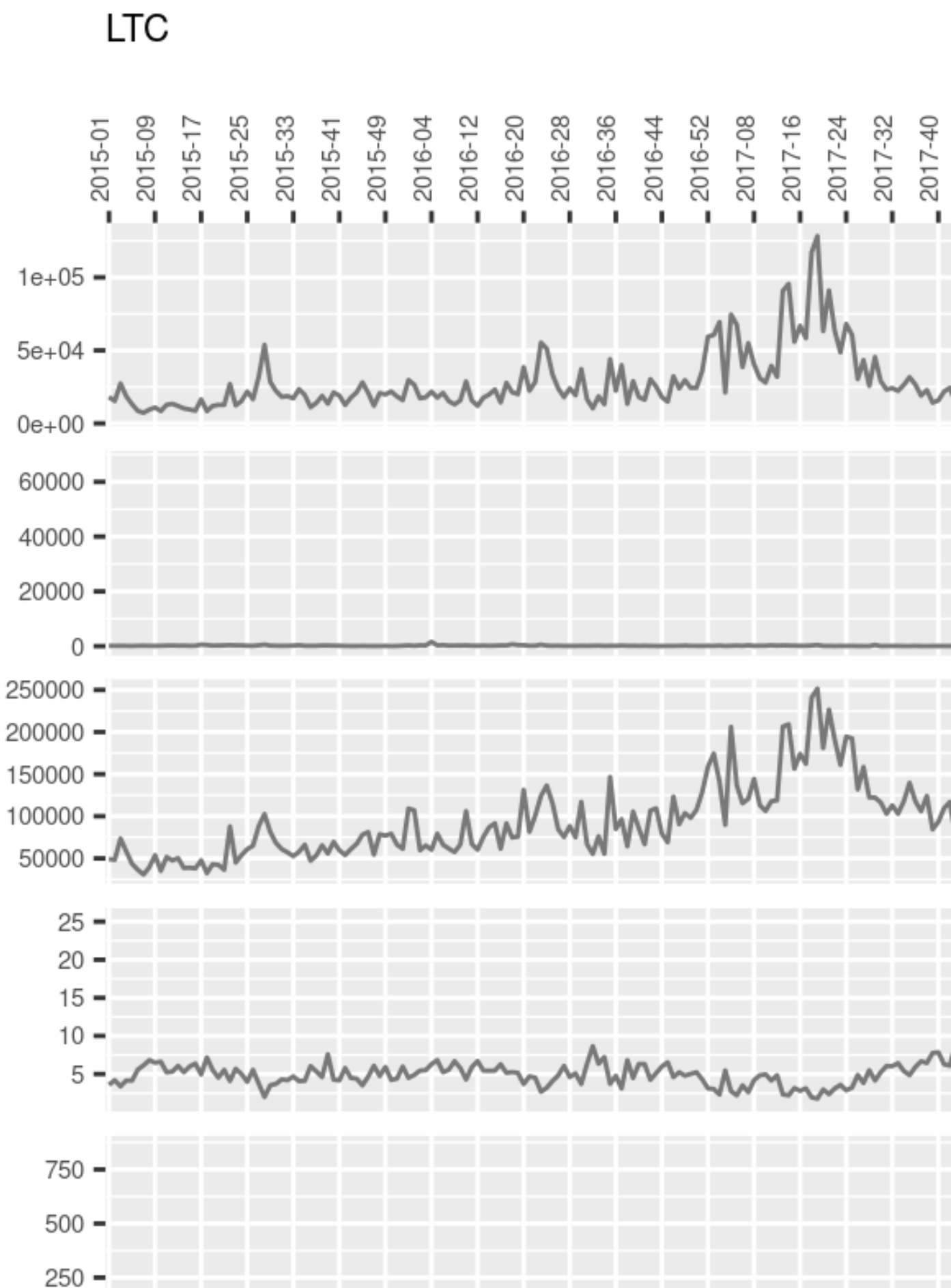


Figure 18: DOGE Summary Characterization of Spend Age Distribution Over Time

25.2 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 6 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 6: 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

25.3 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 7. 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 7: 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

25.4 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

26 Options for Forecasting

My suggestion is to reduce the dimensionality of the forecasting problem by forecasting the parameters of parametric distributions, as in **Section 25.3 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 25.3 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 (29), the final decoy selection algorithm parametric distribution $f(x|\boldsymbol{\beta})$ and parameters $\boldsymbol{\beta}$ should be chosen by:

$$\underset{f(x|\boldsymbol{\beta})\in\mathcal{F}, q\left(\hat{\boldsymbol{\beta}}\right)\in\mathcal{Q}}{\arg\min} \frac{1}{W}\sum_{i=1}^{W} L\left(f\left(x_i \middle| q\left(\hat{\boldsymbol{\beta}}\right)\right)\right), \forall q \in \mathcal{Q} \tag{30}$$

Expression (30) 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 7, Expression (30) would choose `rpln.forecast.accuracy.naive.horizon.period` for BTC, LTC, and DOGE, and `rpln.forecast.accuracy` for BCH. Note that Expression (30) forms a plug-in estimator for the risk function in Equation (18). Therefore, it is a type of empirical risk minimizer.[39]

27 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.

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

References

[Aeeneh et al., 2021] Aeeneh, S., Chervinski, J. O., Yu, J., & Zlatanov, N. (2021). New attacks on the untraceability of transactions in cryptonote-style blockchains. *2021 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, 1–5. https://doi.org/10.1109/ICBC51069.2021.9461130

[Allman et al., 2009] Allman, E. S., Matias, C., & Rhodes, J. A. (2009). Identifiability of parameters in latent structure models with many observed variables. *The Annals of Statistics*, 37(6A), 3099 – 3132. https://doi.org/10.1214/09-AOS689

[Argiento et al., 2020] Argiento, R., Cremaschi, A., & Vannucci, M. (2020). Hierarchical Normalized Completely Random Measures to Cluster Grouped Data. *Journal of the American Statistical Association*, 115(529), 318–333. https://doi.org/10.1080/01621459.2019.159

[Arias-Castro & Jiang, 2021] Arias-Castro, E. & Jiang, H. (2021). *Extending the patra-sen approach to estimating the background component in a two-component mixture model*. https://doi.org/10.48550/ARXIV.2106.13925

[Bagui et al., 2006] Bagui, S., Bagui, S., Chatterjee, A., & Mehra, K. (2006). Classification with multiple independent measurements under a separate sampling scheme. *Statistical Methodology*, 3, 234–251. https://doi.org/10.1016/j.stamet.2005.10.001

[Bassett & Deride, 2019] Bassett, R. & Deride, J. (2019). Maximum a posteriori estimators as a limit of bayes estimators. *Mathematical Programming*, 174(1), 129–144. https://doi.org/10.1007/s10107-018-1241-0

[Benaglia et al., 2009a] Benaglia, T., Chauveau, D., & Hunter, D. R. (2009a). An em-like algorithm for semi- and nonparametric estimation in multivariate mixtures. *Journal of Computational and Graphical Statistics*, 18(2), 505–526. https://doi.org/10.1198/jcgs.2009.07175

[Benaglia et al., 2009b] Benaglia, T., Chauveau, D., Hunter, D. R., & Young, D. S. (2009b). mixtools: An r package for analyzing mixture models. *Journal of Statistical Software*, 32(6), 1–29. https://doi.org/10.18637/jss.v032.i06

[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

[Bonhomme et al., 2016] Bonhomme, S., Jochmans, K., & Robin, J.-M. (2016). Non-parametric estimation of finite mixtures from repeated measurements. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 78(1), 211–229. http://www.jstor.org/stable/24775334

[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

[Copas, 1974] Copas, J. B. (1974). On Symmetric Compound Decision Rules for Dichotomies. *The Annals of Statistics*, 2(1), 199 – 204. https://doi.org/10.1214/aos/1176342626

[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

[Crowder & Hand, 1990] Crowder, M. & Hand, D. (1990). *Analysis of Repeated Measures*. Chapman & Hall/CRC Monographs on Statistics & Applied Probability. Taylor & Francis.

[Deuber et al., 2022] Deuber, D., Ronge, V., & Rueckert, C. (2022). Sok: Assumptions underlying cryptocurrency deanonymizations. *Proceedings on Privacy Enhancing Technologies*, 2022(3). https://petsymposium.org/2022/files/papers/issue3/popets-2022-0091.pdf

[Egger et al., 2022] Egger, C., Lai, R. W. F., Ronge, V., Woo, I. K. Y., & Yin, H. H. F. (2022). On defeating graph analysis of anonymous transactions. *Proceedings on Privacy Enhancing Technologies*, 2022(3). https://petsymposium.org/2022/files/papers/issue3/popets-2022-0085.pdf

[Fix & Hodges, 1989] Fix, E. & Hodges, J. L. (1989). Discriminatory analysis. nonparametric discrimination: Consistency properties. *International Statistical Review / Revue Internationale de Statistique*, 57(3), 238–247. http://www.jstor.org/stable/1403797

[Fraley & Raftery, 2002] Fraley, C. & Raftery, A. E. (2002). Model-based clustering, discriminant analysis, and density estimation. *Journal of the American Statistical Association*, 97(458), 611–631. http://www.jstor.org/stable/3085676

[Frühwirth-Schnatter et al., 2021] Frühwirth-Schnatter, S., Malsiner-Walli, G., & Grün, B. (2021). Generalized Mixtures of Finite Mixtures and Telescoping Sampling. *Bayesian Analysis*, 16(4), 1279 – 1307. https://doi.org/10.1214/21-BA1294

[Hall, 1981] Hall, P. (1981). On the non-parametric estimation of mixture proportions. *Journal of the Royal Statistical Society: Series B (Methodological)*, 43(2), 147–156. https://doi.org/https://doi.org/10.1111/j.2517-6161.1981.tb01164.x

[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/

[Kasahara & Shimotsu, 2014] Kasahara, H. & Shimotsu, K. (2014). Non-parametric identification and estimation of the number of components in multivariate mixtures. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 76(1), 97–111. http://www.jstor.org/stable/24772747

[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

[Kwon & Mbakop, 2021] Kwon, C. & Mbakop, E. (2021). Estimation of the number of components of nonparametric multivariate finite mixture models. *The Annals of Statistics*, 49(4), 2178 – 2205. https://doi.org/10.1214/20-AOS2032

[Levine et al., 2011] Levine, M., Hunter, D. R., & Chauveau, D. (2011). Maximum smoothed likelihood for multivariate mixtures. *Biometrika*, 98(2), 403–416. http://www.jstor.org/stable/23076159

[Lilienfeld & Landfield, 2008] Lilienfeld, S. O. & Landfield, K. (2008). Science and pseudoscience in law enforcement: A user-friendly primer. *Criminal Justice and Behavior*, 35(10), 1215–1230. https://doi.org/10.1177/0093854808321526

[Liu et al., 2019] Liu, C., He, X., Chanyaswad, T., Wang, S., & Mittal, P. (2019). Investigating statistical privacy frameworks from the perspective of hypothesis testing. *Proceedings on Privacy Enhancing Technologies*, 2019(3), 233–254. https://doi.org/10.2478/popets-2019-0045

[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

[Malsiner-Walli et al., 2017] Malsiner-Walli, G., Frühwirth-Schnatter, S., & Grün, B. (2017). Identifying mixtures of mixtures using bayesian estimation. *Journal of Computational and Graphical Statistics*, 26(2), 285–295. https://www.tandfonline.com/doi/full/10.1080/10618600.2016.1200472

[Mbakop, 2017] Mbakop, E. D. (2017). *Identification of auctions with incomplete bid data in the presence of unobserved heterogeneity*. Working Papers, Univ. Calgary. https://cpb-us-e1.wpmucdn.com/sites.northwestern.edu/dist/5/1999/files/2017/10/JMP-Brouillon-1bom42o.pdf

[McLachlan, 1992] McLachlan, G. (1992). *Discriminant Analysis and Statistical Pattern Recognition*. Wiley Series in Probability and Statistics. Wiley.

[McLachlan & Peel, 2000] McLachlan, G. & Peel, D. (2000). *Finite Mixture Models*. Wiley Series in Probability and Statistics. Wiley.

[McLachlan et al., 2019] McLachlan, G. J., Lee, S. X., & Rathnayake, S. I. (2019). Finite mixture models. *Annual Review of Statistics and Its Application*, 6(1), 355–378. https://doi.org/10.1146/annurev-statistics-031017-100325

[Milhaud et al., 2021] Milhaud, X., Pommeret, D., Salhi, Y., & Vandekerkhove, P. (2021). Shape constraint free two-sample contamination model testing. working paper or preprint. https://hal.archives-ouvertes.fr/hal-03201760. working paper or preprint

[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

[Murphy & Topel, 1985] Murphy, K. M. & Topel, R. H. (1985). Estimation and inference in two-step econometric models. *Journal of Business & Economic Statistics*, 3(4), 370–379. https://doi.org/10.1080/07350015.1985.10509471

[Newton, 1996] Newton, M. A. (1996). Bootstrapping phytogenies: Large deviations and dispersion effects. *Biometrika*, 83(2), 315–328. https://www.jstor.org/stable/2337603

[Ni et al., 2021] Ni, W., Cheng, P., Chen, L., & Lin, X. (2021). When the recursive diversity anonymity meets the ring signature. *Proceedings of the 2021 International Conference on Management of Data*, 1359?1371. Association for Computing Machinery. https://doi.org/10.1145/3448016.3452825

[O'Hagan et al., 2012] O'Hagan, A., Murphy, T. B., & Gormley, I. C. (2012). Computational aspects of fitting mixture models via the expectation–maximization algorithm. *Computational Statistics & Data Analysis*, 56(12), 3843–3864. https://www.sciencedirect.com/science/article/pii/S0167947312002101

[Patra & Sen, 2016] Patra, R. K. & Sen, B. (2016). Estimation of a two-component mixture model with applications to multiple testing. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 78(4), 869–893. http://www.jstor.org/stable/24775367

[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

[Ritchie et al., 2020] Ritchie, A., Vandermeulen, R. A., & Scott, C. (2020). Consistent estimation of identifiable nonparametric mixture models from grouped observations. *Advances in Neural Information Processing Systems*, 33, 11676–11686. https://papers.nips.cc/paper/2020/hash/866d90e0921ac7b024b47d672445a086-Abstract.html

[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

[Vandermeulen & Scott, 2019] Vandermeulen, R. A. & Scott, C. D. (2019). An operator theoretic approach to nonparametric mixture models. *The Annals of Statistics*, 47(5), 2704 – 2733. https://doi.org/10.1214/18-AOS1762

[Vankadara et al., 2021] Vankadara, L. C., Bordt, S., von Luxburg, U., & Ghoshdastidar, D. (2021). Recovery guarantees for kernel-based clustering under non-parametric mixture models. *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, 3817–3825. https://proceedings.mlr.press/v130/vankadara21a.html

[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

[Wei & Nguyen, 2022] Wei, Y. & Nguyen, X. (2022). Convergence of de Finetti's mixing measure in latent structure models for observed exchangeable sequences. *The Annals of Statistics*, 50(4), 1859 – 1889. https://doi.org/10.1214/21-AOS2120

[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

[Yu et al., 2019] Yu, J., Au, M. H. A., & Esteves-Verissimo, P. (2019). Re-thinking untraceability in the cryptonote-style blockchain. *2019 IEEE 32nd Computer Security Foundations Symposium (CSF)*, 94–9413. https://doi.org/10.1109/CSF.2019.00014