Software for data-based stochastic programming using bootstrap estimation

Xiaotie Chen
Mathematics Department

David L Woodruff
Graduate School of Management

University of California Davis
Davis CA 95616 USA

August 18, 2022

Abstract

In this paper we describe software for stochastic programming that uses only sampled data to obtain both a consistent sample-average solution and a consistent estimate of confidence intervals for the optimality gap using bootstrap and bagging. The underlying distribution whence the samples come is not required.

1 Introduction

Stochastic programming involves optimization when the input data are uncertain; see, e.g., [KW12] for an introduction. Many research articles on stochastic programming begin with the assumption that uncertain data come from a known distribution. In this paper we describe software that does not rely on this assumption, but uses only sampled data to obtain both a consistent sample-average solution and a consistent estimate of confidence intervals for the optimality gap. The underlying distribution whence the samples come is not required.

The asymptotic properties of solutions as the sample size grows are well known (see [DW88, Sha91] for early examples). For consistent estimators of a confidence interval around the objective function value, we rely on bootstrap and bagging estimators. Consistency and asymptotic normality of the bootstrapped objective function value estimates was proven under fairly general conditions by [ER07]. The use of bagging for such confidence intervals is described in [LQ18]. However, these two papers describe very limited computational experiments. Our paper provides unified notation and a platform for more extensive experimentation based on new open source software that can support on-going research and practice in data-driven stochastic programming.

There have been other applications of bootstrap to stochastic programming. The bootstrap is used to compute confidence intervals for stopping rules for a decomposition algorithm for two-stage stochastic quadratic programs in [LS20] and the authors note that it was used in early versions of the Stochastic Decomposition algorithm [HS91] for stopping rules. There is a technical report [AP11] that discusses some of the theoretical properties of bootstrap samples in the context of stochastic programming. However, their analysis is based on a strong assumption that would not hold for a two-stage problem with inequality constraints, and the analysis does not seem to generalize well to multi-stage problems.

Meanwhile, confidence intervals based on the assumption that the distribution of input data is known in advance have received attention in two main thrusts. The confidence interval for a candidate solution is considered in, e.g., [MMW99, Sha03, CM04, LSW06]. Sequential sampling methods [HS91, BM11, BM09, BPL12], develop stopping rules for sequential sampling procedure that estimate the optimality gap of a sequence of candidate solutions under an increasing sample size.

The open-source software we describe here takes as input an optimization model and a data sample for the uncertain parameters of the optimization model. An additional input user-specified

division of the data for use estimating a solution versus estimating confidence intervals. As output, the software provides an estimated solution, a confidence interval for the value of that solution and a confidence interval for the population solution value. Hence, it provides a mechanism for data-driven optimization of the expected value, which naturally includes expectation-based risk measures such as CVaR.

The paper proceeds as follows. The next section introduces some notation. Section 3 describes the software both in terms of its algorithms and the code. The paper closes with a few conclusions and directions for further research.

2 Notation

We start by introducing some notation that helps explain the software. We have a sample that has been divided into M points for finding an estimated optimal solution and N points for estimating an objective function confidence interval. Alternatively, we could have a sample of N points and be given an estimated solution obtained independently.

- $D = \{d_i,\ i = 1, \ldots, N\}$ is a set of observations from a population Ω. This is the data set that will be used for confidence intervals.
- $F_\Omega(\cdot)$ is the unknown actual distribution function of the population.
- The problem that is our ultimate interest is expressed using

  $$\min_x E_{F_\Omega(d)} g(x, d)$$

  We will consider examples with explicit constraints, but for now we use notation where they are implicit. A simple example of function g is the newsvendor problem: $g(x, d) = c_u(d-x)^+ + c_o(x-d)^+$, where d represents the demand and the cost pair (c_u, c_o) are given as deterministic data. In this case the decision variable x and the (random) data d are scalars, but in general the decision variables and data are vectors. In this paper we never refer to individual elements of the vectors and occasionally use subscripts to refer to particular vectors rather than vector elements.
- x^* is an optimal solution for Ω, so

  $$x^* = \arg\min_x E_{F_\Omega(d)} g(x, d)$$

  which we don't know. The function value w.r.t. the entire population for x^* is

  $$z^* = E_{F_\Omega(d)} g(x^*, d)$$

- $\hat{x}$ is the candidate solution for which we want to estimate the error, and is independent of the set D. We use $\hat{z}_\Omega$ to denote the function value for $\hat{x}$ under the entire population,

  $$\hat{z}_\Omega = E_{F_\Omega(d)} g(\hat{x}, d).$$

  Similarly we define $\hat{z}_N$ as the function value of $\hat{x}$ under observation D of size N,

  $$\hat{z}_N = \frac{1}{N} \sum_{i=1}^{N} g(\hat{x}, d_i)$$

- $\gamma_{gap,\hat{x}}(\Omega)$ is the optimality gap that we are interested in,

  $$\gamma_{gap,\hat{x}}(\Omega) = E_{F_\Omega(d)} g(\hat{x}, d) - E_{F_\Omega(d)} g(x^*, d) = E_{F_\Omega(d)} g(\hat{x}, d) - z^*.$$

  We can regard $\gamma_{gap,\hat{x}}(\cdot)$ as a function that maps a set to the corresponding optimality gap, for example, the optimality gap associated with set D can be represented as

  $$\gamma_{gap,\hat{x}}(D) = \frac{1}{N} \sum_{i=1}^{N} g(\hat{x}, d_i) - \min_x \frac{1}{N} \sum_{i=1}^{N} g(x, d_i)$$

  We drop the subscript $\hat{x}$ when it does not cause confusion, and write the optimality gap function as $\gamma_{gap}(\cdot)$.

3 Software

3.1 Algorithms implemented

We do not describe algorithms for finding a candidate solution $\hat{x}$ because that is already the subject of a large literature. When boot-sp has the task of finding $\hat{x}$, it calls software from mpi-sppy [KMM+21], which calls software from Pyomo [BHH+21].

The algorithms given in this section form the basis for the confidence interval part of software. Given a candidate solution $\hat{x}$, we are interested in deriving a confidence interval for the optimality gap:

$$\gamma_{gap}(\Omega) = E_{F_\Omega(d)} g(\hat{x}, d) - z^*. \tag{1}$$

One may also be interested in deriving a confidence interval for the optimal function value z^*, or for the function value for the candidate solution $E_{F_\Omega(d)} g(\hat{x}, d)$. It turns out that all these estimators can be derived in a similar way as described in the following subsections.

For ease of notation, we assume that we can solve $\min_x \frac{1}{N}\sum_{i=1}^{N} g(x, d_i)$ exactly by simply passing the problem to a solver. We revisit this assumption in Section 5.

3.2 Classical Bootstrap Method

We describe a bootstrap method for estimating the optimality gap as in [ER07]. In the original paper, the bootstrap method is applied to find a confidence interval for the optimal function value z^*, but the same method can be used for optimality gap with minor modification.

At each bootstrap iteration, we resample from set D to get a bootstrap sample $\tilde{D}_b$, and compute the corresponding optimality gap under the set $\tilde{D}_b$,

$$\gamma_{gap}(\tilde{D}_b) = \frac{1}{N}\sum_{\tilde{d}_j \in \tilde{D}_b} g(\hat{x}, \tilde{d}_j) - \min_x \frac{1}{N}\sum_{\tilde{d}_j \in \tilde{D}_b} g(x, \tilde{d}_j).$$

For notational convenience, we introduce $\hat{z}_b = \frac{1}{N}\sum_{\tilde{d}_j \in \tilde{D}_b} g(\hat{x}, \tilde{d}_j)$ as the function value for $\hat{x}$ under set $\tilde{D}_b$, and $z_b^\dagger$ as the optimal function value for $\tilde{D}_b$, so that $z_b^\dagger = \min_x \frac{1}{N}\sum_{\tilde{d}_j \in \tilde{D}_b} g(x, \tilde{d}_j)$, such notation allows us to rewrite the optimality gap under set $\tilde{D}_b$ as

$$\gamma_{gap}(\tilde{D}_b) = \hat{z}_b - z_b^\dagger$$

Similarly we can write empirical optimality gap under the set D as

$$\gamma_{gap}(D) = \frac{1}{N}\sum_{d_j \in D} g(\hat{x}, d_j) - \min_x \frac{1}{N}\sum_{d_j \in D} g(x, d_j) = \hat{z}_N - z_N^\dagger,$$

and the actual optimality gap

$$\gamma_{gap}(\Omega) = \hat{z}_\Omega - z^*$$

It is proven in [ER07] that under certain assumptions, the conditional empirical distribution of $z_b^\dagger - z_N^\dagger$ has the same limit normal distribution as $z_N^\dagger - z^*$ as N tends to infinity, and it follows by the sample average approximation property [Mam12] of $\hat{z}_b$ and $\hat{z}_N$ that $\gamma_{gap}(\tilde{D}_b) - \gamma_{gap}(D)$ has the same limit distribution as $\gamma_{gap}(D) - \gamma_{gap}(\Omega)$. This in turn allows us to derive a confidence interval for the actual optimality gap $\gamma_{gap}(\Omega)$.

Algorithm 1 describes the procedure for finding an approximate confidence interval by using the classical bootstrap procedure.

Algorithm 1: Classical Bootstrap

input : A set D, number of batches B, and a candidate solution $\hat{x}$
Compute $\gamma_{gap}(D) = \hat{z}_N - z_N^\dagger$;
for $b \leftarrow 1$ **to** B **do**
  Resample from D to get set $\tilde{D}_b = \{\tilde{d}_1, \ldots, \tilde{d}_N\}$;
  Compute $\gamma_{gap}(\tilde{D}_b) = \hat{z}_b - z_b^\dagger$,
end
Compute the upper $1-\alpha$-quantile $\varrho_{1-\alpha}$ and lower α-quantile ϱ_α for $\{\gamma_{gap}(\tilde{D}_b) - \gamma_{gap}(D)\}$;
Return $[\gamma_{gap}(D) - \varrho_{1-\alpha}, \gamma_{gap}(D) - \varrho_\alpha]$ as the $(1-2\alpha)$ CI for the optimality gap $\hat{z}_\Omega - z^*$;

If we replace the optimality gap $\gamma_{gap}(D)$ with the optimal function value under set D, i.e. $z_N^\dagger$, and use $z_b^\dagger$ in place of $\gamma_{gap}(\tilde{D}_b)$, then the above algorithm can be used to find an approximate confidence interval for the optimal function value z^*. The same argument applies to all the algorithms in the following subsections.

We also note here that Algorithm 1 can have different variations when different bootstrap procedures are applied. In our software we included a empirical bootstrap procedure as described above, together with a variation that is based on Gaussian approximation.

3.3 Extended Bootstrap Method

Theoretically, the consistency of the estimated confidence interval in Algorithm 1 may no longer stand if the stochastic program does not have a unique solution. An extended two-stage bootstrap method that tackles with such situations is developed in [ER07] and shown in Algorithm 2. Note that the extended bootstrap method is only applicable if we are able to sample from the whole population Ω, which may not be desirable in various examples, but we include the algorithm here for completeness. The consistency for the extended bootstrap method was proven in [ER07] for the CI estimator for the optimal function value, and the result can be easily extended to the optimality gap by noticing the normality of the random variable $z_b^\dagger - z_N^\dagger$ and $z_N^\dagger - z_\Omega^\dagger$. As with the classical bootstrap method, the same algorithm can be used to find a confidence interval for the optimal function value by replacing the optimality gap function $\gamma_{gap}(\cdot)$ with the corresponding empirical optimal function value.

Algorithm 2: Extended Bootstrap

input : A set D, number of batches B, a candidate solution $\hat{x}$

Compute $\gamma_{gap}(D) = \hat{z}_N - z_N^\dagger$;

for $b \leftarrow 1$ **to** B **do**

  Independently sample from Ω to get set $D_b = \{d_1 \ldots, d_N\}$ and compute the corresponding optimality $\gamma_{gap}(D_b)$;

  Resample from D_b to get $\tilde{D}_b = \{\tilde{d}_1, \ldots, \tilde{d}_N\}$ and compute $\gamma_{gap}(\tilde{D}_b)$;

end

Compute the upper $1-\alpha$-quantile $\varrho_{1-\alpha}$ and lower α-quantile ϱ_α for $\{\gamma_{gap}(\tilde{D}_b) - \gamma_{gap}(D_b)\}$;

Independently sample from Ω to get $\bar{D}_N$ and $\check{D}_N$, each of size N ;

Compute the center of the confidence interval $G = 2 * \gamma_{gap}(\bar{D}_N \cup \check{D}_N) - \gamma_{gap}(\bar{D}_N)$;

Return $[G - \varrho_{1-\alpha}, G - \varrho_\alpha]$ as the $(1-2\alpha)$ CI for the optimality gap $\hat{z}_\Omega - z^*$;

3.4 Subsampling

The subsampling method for estimating the confidence interval of z^* was briefly mentioned in [ER07]. The procedure is essentially the same as the classical bootstrap method, except that a smaller subsampling size is adopted for resampling. Algorithm 3 details the procedure for finding the confidence interval of the optimality gap, and the consistency of the estimator is proven in [ER07].

Algorithm 3: Subampling

input : A set D, number of batches B, sub-sample size k, and a candidate solution $\hat{x}$

Compute $\gamma_{gap}(D) = \hat{z}_N - z_N^\dagger$;

for $b \leftarrow 1$ **to** B **do**

  Resample from D without replacement to get set $\tilde{D}_b$ of size k, $\tilde{D}_b = \{\tilde{d}_1, \ldots, \tilde{d}_k\}$;

  Compute $\gamma_{gap}(\tilde{D}_b)$,

end

Compute the upper $1-\alpha$-quantile $\varrho_{1-\alpha}$ and lower α-quantile ϱ_α for $\{\sqrt{k}(\gamma_{gap}(\tilde{D}_b) - \gamma_{gap}(D))\}$;

Return $[\gamma_{gap}(D) - \sqrt{1/N}\varrho_{1-\alpha}, \gamma_{gap}(D) - \sqrt{1/N}\varrho_\alpha]$ as the $(1-2\alpha)$ CI for the optimality gap $\hat{z}_\Omega - z^*$;

3.5 Bagging

A bagging procedure proposed in [LQ18] approximates the confidence interval from the optimality gap without the need to sample for the entire population. We note here that the bagging procedure described in [LQ18] bears some similarity with the subsampling method in Algorithm 3, as in both cases a smaller subsampling size is used. The difference lies in how the confidence interval is formed: the subsampling method directly uses the quantiles of the subsamples to derive a confidence interval, whereas the bagging procedure uses an empirical version of the infinitesimal jackknife estimator for variance estimation.

Algorithm 4 illustrates the procedure in using bagging to find an approximate confidence interval for the optimality gap. The original algorithm in [LQ18] focuses on finding CI for the optimal function value, and Algorithm 4 is a small variation of the original one.

Algorithm 4: Bagging-based sampling

input : A set D, number of bags B, bag sample size k, and a candidate solution $\hat{x}$

Compute $\gamma_{gap}(D) = z_N^\dagger - \hat{z}_N$;

for $b \leftarrow 1$ **to** B **do**

  Resample from D to get bagging set $\tilde{D}_b$ of size k, $\tilde{D}_b = \{\tilde{d}_1, \ldots, \tilde{d}_k\}$;

  Compute $\gamma_{gap}(\tilde{D}_b)$,

end

Compute the mean of $\gamma_{gap}(\tilde{D}_b)$ as the center of the confidence interval, so

$$G = \frac{1}{B}\sum_{b=1}^{B} \gamma_{gap}(\tilde{D}_b)$$

Compute the error term

$$\tilde{\sigma}^2 = \begin{cases} \sum_{i=1}^n \widehat{cov}_i^2 & \text{if with replacement} \\ \frac{n^2}{(n-k)^2}\sum_{i=1}^n \widehat{cov}_i^2 & \text{if without replacement} \end{cases},$$

where

$$\widehat{cov}_i^2 = \frac{1}{B}\sum_{b=1}^{B}(N_i^b - k/n)(\gamma_{gap}(\tilde{D}_b) - G),$$

and $N_i^b = \#\{j : \tilde{d}_j = d_i\}$;

Return $[G - \delta_{1-\alpha}\tilde{\sigma}, G + \delta_{1-\alpha}\tilde{\sigma}]$ as the $(1-2\alpha)$ CI for the optimality gap $\hat{z}_\Omega - z^*$, with $\delta_{1-\alpha}$ being the $(1-\alpha)$ quantile for a standard normal variable;

3.6 Code

The github site https://github.com/boot-sp/boot-sp.git provides a Python implementation of software called BOOT-SP that computes solutions with confidence intervals and also supports simulations for research purposes. A user of the software provides a Python module that has a few procedures, the most important of which takes a sample of data, d, as an argument and returns a Pyomo model for $g(x, d)$. The module also contains a helper procedures to deal with data processing related to the particular problem.

The algorithms give rise to the following methods supported by the code:

- Classical_gaussian,
- Classical_quantile,
- Extended,
- Subsampling,
- Bagging_with_replacement, and
- Bagging_without_replacement.

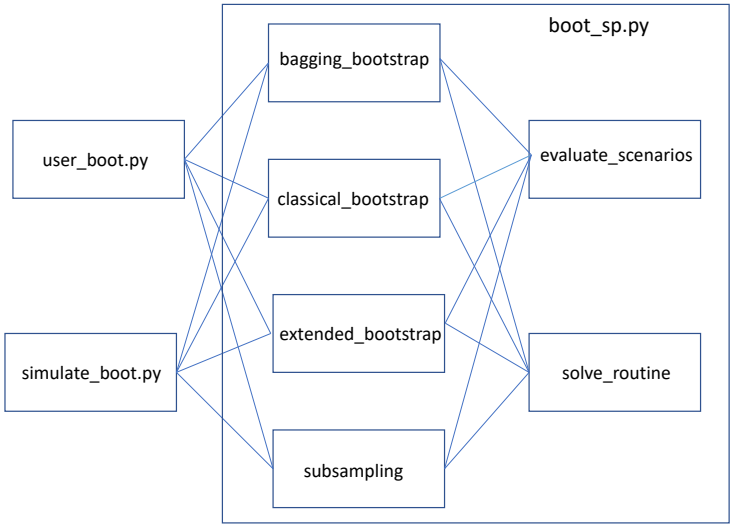

Figure 1: High level view of software organization.

Figure 1 gives an overview of the top-level software components. For the classical bootstrap, the use of the Gaussian or quantiles is controlled by parameters as is bagging with, and without, replacement. We are able to exploit the similarities between the methods by creating parameterized functions that do the calculations. The function `solve_routine` calls lower-level code in `mpi-sppy` and `Pyomo` to minimize functions of the form $\frac{1}{N}\sum_{d_j \in D} g(x, d_j$. The function `solve_routine` calls lower-level code in `mpi-sppy` and `Pyomo` to evaluate expectations of the form such as $\frac{1}{N}\sum_{d_j \in D} g(\hat{x}, d_j$. The differences in methods are due to the way in which these functions are used.

The algorithms given in Section 3.1 have loops over B. In `boot-sp`, those loops are parallelized using MPI. MPI offers advantages for distributed memory computers at the expense of being difficult to install, particularly on Windows computers.

In most serious applications, we expect that `boot_sp.py` will be used as a callable library. Nonetheless, the `user_boot.py` program is provided to demonstrate the concept of data-driven stochastic programming so it provides full command line arguments. The main contribution of `boot-sp` is bootstrap or bagging esimation of confidence intervals. Consequently, it is expected the users will have other code to find a candidate solution $\hat{x}$. However, in order for `boot-sp` to give a concrete illustration of the concept of data-driving stochastic programming we offer an option to call a function that uses `mpi-sppy` to compute $\hat{x}$.

The `simulate_boot.py` software is callable from the command line, but that is mainly for illustration purposes. We also distribute the program `simulate_experiments.py` that runs the experiments described in Section 4. It calls `simulate_boot.py` in a loop to compute coverage and width statistics for the confidence intervals.

4 Experiments

We demonstrate experiments on the algorithms over a few examples as detailed in the following subsections. For simulation purpose, we grant ourselves the access to the distribution of the entire population F_Ω, and approximate the theoretical optimal function value z^* by drawing an extremely large number of samples from the distribution F_Ω, then compute the corresponding optimal function value, which we use as z^* for the simulations

We replicate the execution of each algorithm a large number (e.g., 500) times with a fixed candidate solution $\hat{x}$. At each replication, a new set D is drawn independently from F_Ω, and the algorithm is executed to return a $(1-2\alpha)$ confidence interval for the optimal function value z^*. The coverage rate is then reported as the percentages of the confidence intervals that contains z^*

over the 500 runs. We also report the average length of the confidence intervals, denoted as "avg len", over the repeated experiments for each algorithm. The average length of the CIs to some extent represents the sharpness of the intervals computed by the algorithms.

4.1 Small Schultz Examples

4.1.1 Unique Solution

This examples is from [SSVDV98] used in [ER07, p. 129]. Results are shown in Table 1.

4.1.2 Nonunique Solution

This example is a modified version of the previous problem that used in [ER07, p. 131]. Results are shown in Table 2. This problem has multiple optima.

4.2 CVaR

A one-stage CVaR problem is used by [LQ18]:

$$\min_x \left\{ x + \frac{1}{a} E\left[(\xi - x)_+\right] \right\}$$

where $(\cdot)_+$ is defined as $\max \cdot, 0$, $a = 0.1$ and ξ is a drawn from a standard normal distribution. Results are shown in Table 3.

4.3 Scalable Farmer

This example is based on the well-known farmer example from [BL97] as modifed for stress-testing various pieces of software such as [KMM+21]. To make it scalable, two instance creation parameters *cropsmult* and *numscens* are added. The original problem has three crops and three scenarios. The scalable instances have *cropsmult* sets of the original three crops with the characteristics as in the original problem and yields that depend on the scenario. Scenarios are in groups of three with a uniformly distributed psuedo-random number added to the yield values of the original three scenarios.

For the results shown in Tables 4 and 5, we used the original three crops and 1000 scenarios to get an assumed value for z^*.

4.4 Discussion of Results

These experiments are intended mainly to illustrate that the software can be used for such experiments. They do illustrate the unsurprising result that if the samples are too small, the confidence intervals will not be very good. They also suggest that the method we call Extended, which is sort of an afterthought in [ER07], does not seem to work all that well.

The results are mostly reasonable, but mixed and depend on the availability of enough data as well as the choice of method and parameters. Detailed conclusions are beyond the scope of this small study. One other thing to note, though, concerns CVaR. Since CVaR considers the tail, getting good confidence intervals requires a larger value of N. Perhaps for similar reasons, quantile-based intervals do not seem to be as good as Gausian.

5 Conclusions and Directions for Further Research

This paper describes software support for the idea that stochastic programming need not be based on an assumption that data come from a known distribution. Clearly, sampled data can be used to obtain an estimated solution in a way that is asymptotically valid. The main thing that is new here is that we have created software that also uses sampled data to estimate the quality of that solution in a way that is asymptotically valid. We now describe a sample from opportunities for further work.

Throughout the paper, for notational convenience we have worked with exact optimizations for samples of size N, but with more notation any method (such as a decomposition) that can get upper and lower bounds could be used. This gives rise to empirical questions about the interaction between the tightness of the bounds and the quality of the confidence intervals.

Another experimental question concerns the common problem of allocation of the finite samples: what fraction should go to computing $\hat{x}$ and what fraction should go to estimation of confidence

method	N	nB	k	avg len	coverage
Classical_gaussian	40	100	0	8.04	0.92
Classical_gaussian	80	100	0	5.58	0.90
Classical_gaussian	40	500	0	8.01	0.92
Classical_gaussian	80	500	0	5.73	0.93
Classical_quantile	40	100	0	7.82	0.88
Classical_quantile	80	100	0	5.41	0.87
Classical_quantile	40	500	0	7.97	0.90
Classical_quantile	80	500	0	5.69	0.89
Bagging_with_replacement	40	100	16	9.36	0.93
Bagging_with_replacement	40	100	24	9.38	0.95
Bagging_with_replacement	40	100	32	9.42	0.96
Bagging_with_replacement	80	100	32	7.62	0.97
Bagging_with_replacement	80	100	48	7.39	0.97
Bagging_with_replacement	80	100	64	7.49	0.98
Bagging_with_replacement	40	500	16	8.21	0.90
Bagging_with_replacement	40	500	24	8.15	0.89
Bagging_with_replacement	40	500	32	8.26	0.91
Bagging_with_replacement	80	500	32	6.08	0.93
Bagging_with_replacement	80	500	48	6.05	0.95
Bagging_with_replacement	80	500	64	6.07	0.95
Bagging_without_replacement	40	100	16	9.32	0.95
Bagging_without_replacement	40	100	24	9.59	0.93
Bagging_without_replacement	40	100	32	9.43	0.94
Bagging_without_replacement	80	100	32	7.54	0.98
Bagging_without_replacement	80	100	48	7.83	0.98
Bagging_without_replacement	80	100	64	7.68	0.95
Bagging_without_replacement	40	500	16	8.42	0.91
Bagging_without_replacement	40	500	24	8.58	0.94
Bagging_without_replacement	40	500	32	8.60	0.93
Bagging_without_replacement	80	500	32	6.16	0.93
Bagging_without_replacement	80	500	48	6.29	0.95
Bagging_without_replacement	80	500	64	6.26	0.95
Subsampling	40	100	16	6.15	0.79
Subsampling	40	100	24	5.08	0.73
Subsampling	40	100	32	3.52	0.58
Subsampling	80	100	32	4.36	0.80
Subsampling	80	100	48	3.62	0.72
Subsampling	80	100	64	2.54	0.55
Subsampling	40	500	16	6.30	0.81
Subsampling	40	500	24	5.18	0.74
Subsampling	40	500	32	3.65	0.56
Subsampling	80	500	32	4.45	0.82
Subsampling	80	500	48	3.68	0.74
Subsampling	80	500	64	2.58	0.56
Extended	40	100	0	8.29	0.88
Extended	80	100	0	5.37	0.81
Extended	40	500	0	8.22	0.90
Extended	80	500	0	5.60	0.86

Table 1: Results for unique_schultz with α=0.05 based on 100 replications.

method	N	nB	k	avg len	coverage
Classical_gaussian	40	100	0	7.92	0.90
Classical_gaussian	80	100	0	5.46	0.91
Classical_gaussian	40	500	0	7.89	0.89
Classical_gaussian	80	500	0	5.60	0.93
Classical_quantile	40	100	0	7.72	0.89
Classical_quantile	80	100	0	5.30	0.86
Classical_quantile	40	500	0	7.84	0.90
Classical_quantile	80	500	0	5.57	0.92
Bagging_with_replacement	40	100	16	9.23	0.93
Bagging_with_replacement	40	100	24	9.24	0.94
Bagging_with_replacement	40	100	32	9.28	0.96
Bagging_with_replacement	80	100	32	7.45	0.96
Bagging_with_replacement	80	100	48	7.23	0.97
Bagging_with_replacement	80	100	64	7.32	0.97
Bagging_with_replacement	40	500	16	8.10	0.89
Bagging_with_replacement	40	500	24	8.02	0.89
Bagging_with_replacement	40	500	32	8.13	0.91
Bagging_with_replacement	80	500	32	5.95	0.92
Bagging_with_replacement	80	500	48	5.92	0.94
Bagging_with_replacement	80	500	64	5.94	0.94
Bagging_without_replacement	40	100	16	9.19	0.95
Bagging_without_replacement	40	100	24	9.43	0.93
Bagging_without_replacement	40	100	32	9.27	0.92
Bagging_without_replacement	80	100	32	7.38	0.98
Bagging_without_replacement	80	100	48	7.63	0.98
Bagging_without_replacement	80	100	64	7.50	0.96
Bagging_without_replacement	40	500	16	8.29	0.90
Bagging_without_replacement	40	500	24	8.44	0.91
Bagging_without_replacement	40	500	32	8.46	0.93
Bagging_without_replacement	80	500	32	6.03	0.93
Bagging_without_replacement	80	500	48	6.14	0.96
Bagging_without_replacement	80	500	64	6.11	0.94
Subsampling	40	100	16	6.04	0.82
Subsampling	40	100	24	4.99	0.72
Subsampling	40	100	32	3.45	0.61
Subsampling	80	100	32	4.28	0.81
Subsampling	80	100	48	3.52	0.76
Subsampling	80	100	64	2.48	0.57
Subsampling	40	500	16	6.21	0.83
Subsampling	40	500	24	5.10	0.78
Subsampling	40	500	32	3.59	0.60
Subsampling	80	500	32	4.36	0.83
Subsampling	80	500	48	3.60	0.78
Subsampling	80	500	64	2.52	0.56
Extended	40	100	0	8.16	0.86
Extended	80	100	0	5.16	0.81
Extended	40	500	0	8.08	0.88
Extended	80	500	0	5.45	0.89

Table 2: Results for nonunique_schultz with α=0.05 based on 100 replications.

method	N	nB	k	avg len	coverage
Classical_gaussian	300	100	0	0.36	0.90
Classical_gaussian	600	100	0	0.26	0.82
Classical_gaussian	300	1000	0	0.36	0.94
Classical_gaussian	600	1000	0	0.26	0.80
Classical_quantile	300	100	0	0.35	0.50
Classical_quantile	600	100	0	0.25	0.57
Classical_quantile	300	1000	0	0.36	0.69
Classical_quantile	600	1000	0	0.26	0.58
Bagging_with_replacement	300	100	120	0.63	1.00
Bagging_with_replacement	300	100	180	0.62	1.00
Bagging_with_replacement	300	100	240	0.62	1.00
Bagging_with_replacement	600	100	240	0.62	1.00
Bagging_with_replacement	600	100	360	0.63	1.00
Bagging_with_replacement	600	100	480	0.62	1.00
Bagging_with_replacement	300	1000	120	0.20	1.00
Bagging_with_replacement	300	1000	180	0.20	1.00
Bagging_with_replacement	300	1000	240	0.20	1.00
Bagging_with_replacement	600	1000	240	0.20	1.00
Bagging_with_replacement	600	1000	360	0.20	1.00
Bagging_with_replacement	600	1000	480	0.20	1.00
Bagging_without_replacement	300	100	120	0.81	1.00
Bagging_without_replacement	300	100	180	0.99	1.00
Bagging_without_replacement	300	100	240	1.39	1.00
Bagging_without_replacement	600	100	240	0.80	1.00
Bagging_without_replacement	600	100	360	0.99	1.00
Bagging_without_replacement	600	100	480	1.39	1.00
Bagging_without_replacement	300	1000	120	0.26	1.00
Bagging_without_replacement	300	1000	180	0.31	1.00
Bagging_without_replacement	300	1000	240	0.45	1.00
Bagging_without_replacement	600	1000	240	0.26	1.00
Bagging_without_replacement	600	1000	360	0.32	1.00
Bagging_without_replacement	600	1000	480	0.45	1.00
Subsampling	300	100	120	0.36	0.65
Subsampling	300	100	180	0.35	0.61
Subsampling	300	100	240	0.35	0.60
Subsampling	600	100	240	0.26	0.64
Subsampling	600	100	360	0.26	0.60
Subsampling	600	100	480	0.25	0.50
Subsampling	300	1000	120	0.36	0.67
Subsampling	300	1000	180	0.36	0.65
Subsampling	300	1000	240	0.36	0.62
Subsampling	600	1000	240	0.26	0.63
Subsampling	600	1000	360	0.26	0.62
Subsampling	600	1000	480	0.26	0.67
Extended	300	100	0	0.50	0.79
Extended	600	100	0	0.36	0.71
Extended	300	1000	0	0.51	0.85
Extended	600	1000	0	0.36	0.81

Table 3: Results for cvar with α=0.05 based on 100 replications.

method	N	nB	k	avg len	coverage
Classical_gaussian	30	100	0	28259.46	0.887
Classical_gaussian	60	100	0	20181.52	0.892
Classical_gaussian	30	1000	0	28294.08	0.885
Classical_gaussian	60	1000	0	20272.41	0.907
Classical_quantile	30	100	0	27343.29	0.870
Classical_quantile	60	100	0	19645.95	0.877
Classical_quantile	30	1000	0	28236.30	0.880
Classical_quantile	60	1000	0	20235.36	0.905
Bagging_with_replacement	30	100	12	31422.27	0.932
Bagging_with_replacement	30	100	18	31257.34	0.938
Bagging_with_replacement	30	100	24	32061.31	0.932
Bagging_with_replacement	60	100	24	25164.00	0.955
Bagging_with_replacement	60	100	36	25298.68	0.953
Bagging_with_replacement	60	100	48	25285.28	0.950
Bagging_with_replacement	30	1000	12	28323.03	0.895
Bagging_with_replacement	30	1000	18	28489.61	0.902
Bagging_with_replacement	30	1000	24	28554.10	0.900
Bagging_with_replacement	60	1000	24	20614.13	0.907
Bagging_with_replacement	60	1000	36	20688.71	0.907
Bagging_with_replacement	60	1000	48	20751.51	0.905
Bagging_without_replacement	30	100	12	32638.99	0.935
Bagging_without_replacement	30	100	18	32640.54	0.920
Bagging_without_replacement	30	100	24	32558.19	0.930
Bagging_without_replacement	60	100	24	25546.72	0.965
Bagging_without_replacement	60	100	36	25369.97	0.963
Bagging_without_replacement	60	100	48	25711.17	0.970
Bagging_without_replacement	30	1000	12	29438.81	0.902
Bagging_without_replacement	30	1000	18	29629.62	0.905
Bagging_without_replacement	30	1000	24	29664.24	0.895
Bagging_without_replacement	60	1000	24	21069.04	0.915
Bagging_without_replacement	60	1000	36	21175.03	0.917
Bagging_without_replacement	60	1000	48	21287.06	0.922
Subsampling	30	100	12	21560.28	0.777
Subsampling	30	100	18	17703.34	0.680
Subsampling	30	100	24	12593.48	0.502
Subsampling	60	100	24	15389.09	0.785
Subsampling	60	100	36	12457.75	0.680
Subsampling	60	100	48	8943.31	0.500
Subsampling	30	1000	12	22124.33	0.797
Subsampling	30	1000	18	18209.22	0.690
Subsampling	30	1000	24	12877.64	0.510
Subsampling	60	1000	24	15739.58	0.790
Subsampling	60	1000	36	12903.63	0.700
Subsampling	60	1000	48	9166.89	0.525
Extended	30	100	0	26890.50	0.848
Extended	60	100	0	19620.82	0.863
Extended	30	1000	0	28080.65	0.870
Extended	60	1000	0	20435.31	0.875

Table 4: Results for farmer based on 400 replications.

method	N	nB	k	avg len	coverage
Classical_gaussian	30	100	0	21809.65	0.750
Classical_gaussian	60	100	0	15622.85	0.740
Classical_gaussian	30	1000	0	21955.60	0.770
Classical_gaussian	60	1000	0	15648.47	0.730
Classical_quantile	30	100	0	21348.92	0.740
Classical_quantile	60	100	0	15337.17	0.740
Classical_quantile	30	1000	0	22030.79	0.740
Classical_quantile	60	1000	0	15641.96	0.740
Bagging_with_replacement	30	100	12	24013.20	0.800
Bagging_with_replacement	30	100	18	24468.50	0.790
Bagging_with_replacement	30	100	24	25117.26	0.830
Bagging_with_replacement	60	100	24	19617.59	0.830
Bagging_with_replacement	60	100	36	19544.47	0.820
Bagging_with_replacement	60	100	48	19788.37	0.860
Bagging_with_replacement	30	1000	12	21923.88	0.730
Bagging_with_replacement	30	1000	18	22105.87	0.760
Bagging_with_replacement	30	1000	24	22159.09	0.770
Bagging_with_replacement	60	1000	24	16040.28	0.750
Bagging_with_replacement	60	1000	36	16049.88	0.760
Bagging_with_replacement	60	1000	48	16071.97	0.760
Bagging_without_replacement	30	100	12	25178.82	0.830
Bagging_without_replacement	30	100	18	25316.79	0.780
Bagging_without_replacement	30	100	24	25361.81	0.790
Bagging_without_replacement	60	100	24	19903.04	0.890
Bagging_without_replacement	60	100	36	19887.07	0.860
Bagging_without_replacement	60	100	48	19734.46	0.880
Bagging_without_replacement	30	1000	12	22641.34	0.760
Bagging_without_replacement	30	1000	18	23047.99	0.780
Bagging_without_replacement	30	1000	24	23065.88	0.770
Bagging_without_replacement	60	1000	24	16416.91	0.750
Bagging_without_replacement	60	1000	36	16428.75	0.780
Bagging_without_replacement	60	1000	48	16453.14	0.760
Subsampling	30	100	12	17165.90	0.700
Subsampling	30	100	18	13965.11	0.550
Subsampling	30	100	24	9980.19	0.400
Subsampling	60	100	24	12253.03	0.640
Subsampling	60	100	36	9822.92	0.520
Subsampling	60	100	48	6949.98	0.370
Subsampling	30	1000	12	17245.27	0.670
Subsampling	30	1000	18	14230.53	0.550
Subsampling	30	1000	24	10112.13	0.410
Subsampling	60	1000	24	12336.30	0.630
Subsampling	60	1000	36	10034.51	0.510
Subsampling	60	1000	48	7141.78	0.350
Extended	30	100	0	21392.68	0.760
Extended	60	100	0	16125.62	0.730
Extended	30	1000	0	21603.71	0.770
Extended	60	1000	0	16038.97	0.780

Table 5: Results for farmer with α=0.1 based on 100 replications.

intervals. Another thing to explore is that there is no requirement $\hat{x}$ be obtained by solving a problem of minimizing expected value. Other models, such as Robust Optimization (see, e.g., [BTN02]) or Distributionally Robust Optimization (see, e.g., [VPEK21]),might be used, but there may still be an interest in using sampled data to evaluate confidence intervals for the expected value and conditional value at risk.

References

[AP11] Mihai Anitescu and C Petra. Higher-order confidence intervals for stochastic programming using bootstrapping. Technical report, Citeseer, 2011.

[BHH+21] Michael L. Bynum, Gabriel A. Hackebeil, William E. Hart, Carl D. Laird, Bethany L. Nicholson, John D. Siirola, Jean-Paul Watson, and David L. Woodruff. *Pyomo–optimization modeling in python*, volume 67. Springer Science & Business Media, third edition, 2021.

[BL97] John. R. Birge and François Louveaux. *Introduction to Stochastic Programming*. Springer Verlag, 1997.

[BM09] Güzin Bayraksan and David P Morton. Assessing solution quality in stochastic programs via sampling. In *Decision Technologies and Applications*, pages 102–122. Informs, 2009.

[BM11] Güzin Bayraksan and David P Morton. A sequential sampling procedure for stochastic programming. *Operations Research*, 59(4):898–913, 2011.

[BPL12] Guzin Bayraksan and Péguy Pierre-Louis. Fixed-width sequential stopping rules for a class of stochastic programs. *SIAM Journal on Optimization*, 22(4):1518–1548, 2012.

[BTN02] Aharon Ben-Tal and Arkadi Nemirovski. Robust optimization–methodology and applications. *Mathematical programming*, 92(3):453–480, 2002.

[CM04] Anukal Chiralaksanakul and David P Morton. *Assessing policy quality in multi-stage stochastic programming*. Humboldt-Universität zu Berlin, Mathematisch-Naturwissenschaftliche Fakultät ..., 2004.

[DW88] Jitka Dupacová and Roger Wets. Asymptotic behavior of statistical estimators and of optimal solutions of stochastic optimization problems. *The annals of statistics*, 16(4):1517–1549, 1988.

[ER07] Andreas Eichhorn and Werner Römisch. Stochastic integer programming: Limit theorems and confidence intervals. *Mathematics of Operations Research*, 32(1):118–135, 2007.

[HS91] Julia L Higle and Suvrajeet Sen. Stochastic decomposition: An algorithm for two-stage linear programs with recourse. *Mathematics of operations research*, 16(3):650–669, 1991.

[KMM+21] Bernard Knueven, David Mildebrath, Christopher Muir, John D Siirola, Jean-Paul Watson, and David L Woodruff. A parallel hub-and-spoke system for large-scale scenario-based optimization under uncertainty. *https://mpi-sppy.readthedocs.io/en/latest/*, 2021.

[KW12] Alan J. King and Stein W. Wallace. *Modeling with Stochastic Programming*. Springer, 2012.

[LQ18] Henry Lam and Huajie Qian. Assessing solution quality in stochastic optimization via bootstrap aggregating. In *2018 Winter Simulation Conference (WSC)*, pages 2061–2071. IEEE, 2018.

[LS20] Junyi Liu and Suvrajeet Sen. Asymptotic results of stochastic decomposition for two-stage stochastic quadratic programming. *SIAM Journal on Optimization*, 30(1):823–852, 2020.

[LSW06] Jeff Linderoth, Alexander Shapiro, and Stephen Wright. The empirical behavior of sampling methods for stochastic programming. *Annals of Operations Research*, 142(1):215–241, 2006.

[Mam12] Enno Mammen. *When does bootstrap work?: asymptotic results and simulations*, volume 77. Springer Science & Business Media, 2012.

[MMW99] Wai-Kei Mak, David P Morton, and R Kevin Wood. Monte carlo bounding techniques for determining solution quality in stochastic programs. *Operations research letters*, 24(1-2):47–56, 1999.

[Sha91] Alexander Shapiro. Asymptotic analysis of stochastic programs. *Annals of Operations Research*, 30(1):169–186, 1991.

[Sha03] Alexander Shapiro. Inference of statistical bounds for multistage stochastic programming problems. *Mathematical Methods of Operations Research*, 58(1):57–68, 2003.

[SSVDV98] Rüdiger Schultz, Leen Stougie, and Maarten H Van Der Vlerk. Solving stochastic programs with integer recourse by enumeration: A framework using gröbner basis. *Mathematical Programming*, 83(1):229–252, 1998.

[VPEK21] Bart PG Van Parys, Peyman Mohajerin Esfahani, and Daniel Kuhn. From data to decisions: Distributionally robust optimization is optimal. *Management Science*, 67(6):3387–3402, 2021.