Technomics
Better Decisions Faster

Risky Business: Navigating the World of Software Productivity

Dave Brown, Subject Matter Expert, Technomics, DBrown@technomics.net

Kevin Cincotta, Principal Cost Analyst, The MITRE Corporation, KCincotta@mitre.org

International Cost Estimating and Analysis Association (ICEAA) Professional Development & Training Workshop

May 2024

Abstract

Size and productivity are commonly cited as the two major software development cost drivers. Logic dictates that the two are related and inversely correlated. But what is the probabilistic range of uncertainty for productivity, *given* a software size? What is meant by "an 80% confidence level for productivity"? Cost analysts often quantify uncertainty with an S-Curve; why can't this be done for productivity directly? We use International Software Benchmarking Standards Group (ISBSG) data to estimate the distribution of productivity directly and provide closed-form formulas for the fitted distribution(s). We find that productivity (and, with certain assumptions, cost) can be estimated with an S-Curve directly, using built-in Excel formulas, with no need for Monte Carlo simulation. This result has significant implications for almost any software development cost estimate and is particularly relevant to agile development efforts where time-boxed effort is generally fixed.

Keywords: *Agile, Risk, Software, Uncertainty, Productivity, Probability Distribution, S-Curve, Monte Carlo*

Table of Contents

Introduction

Software productivity is an important concept in any software development estimate. It is sometimes specified as a direct input and sometimes included implicitly through other inputs. Regardless of how it is treated, the estimating community will benefit from a better understanding of the risk and uncertainty associated with productivity. A complete analysis of this topic requires that estimators understand risk and uncertainty within the components of productivity, productivity calculations, as well as the final software development estimate.

This paper evaluates the most common methods estimators currently use to address productivity, and presents an alternative approach that will:

- Improve the accuracy of software development estimates by presenting an alternative way of incorporating productivity.
- Improve the cost risk and uncertainty analysis of software development estimates by better modeling productivity uncertainty.

Background

We begin with a discussion of the key concepts associated with productivity. We then explore the most used methods for including it in software development estimates.

Software Productivity as a Concept

Along with size, productivity is one of the two main cost drivers for software development. Software size measures the quantity of software that must be developed. Productivity measures the efficiency of development, which is driven by software complexity as well as development team capability.

A more general definition of productivity is the ratio of the effort associated with unit of output to a unit of input. This concept applies in the world of software development. If we wish to define productivity using this general concept, we consider output divided by input. For example, software size (measured in complexity-Adjusted Function Points

(AFP)) divided by productivity (measured in AFP per man-hour) will result in an effort estimate measured in man-hours.

All the different methods of measuring productivity used by the cost community share the general concept of measuring the ratio of output to input. However, estimators sometimes measure productivity as effort divided by size. For example, project delivery rate (PDR) measures man-hours per software sizing metric. In this case, PDR is the multiplicative inverse (reciprocal) of our standard productivity definition and used in effort estimation by multiplying size by PDR.

In this paper, we define productivity as size divided by effort. This is consistent with the general definition and follows an intuitive principle that higher productivity should result in less effort and therefore less cost.

Methods for Estimating Software Productivity

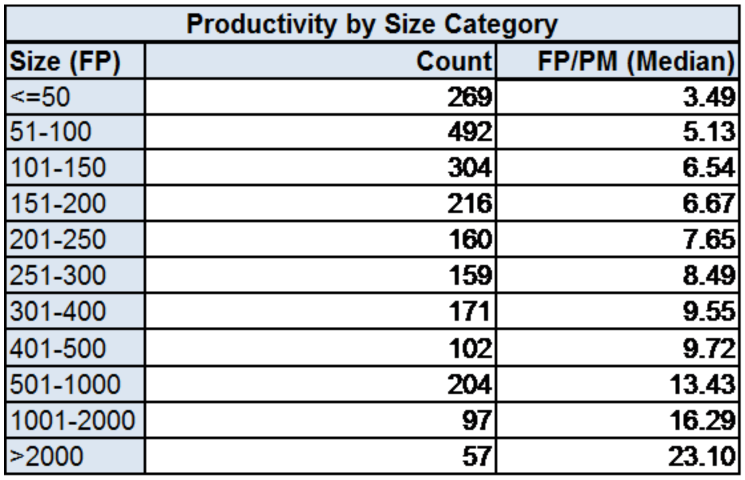
Just as there are many ways to estimate software development, there are many ways to measure and estimate productivity. These methods can be grouped into two logical categories: productivity as an input and productivity as an output.

Productivity as an Input

When productivity is measured as an input, it is used in an estimating equation as an input variable. The most common examples of productivity as an input include:

- Assumed factor. In this method, the estimator uses a published productivity factor. For example, if a function point count of 350 Function Points (FP) is obtained, then the estimator might use the following figure[1] to obtain productivity:

[1] Beckett, Donald, *An Analysis of Function Point Trends*, https://www.qsm.com/articles/analysis-function-point-trends



Productivity by Size Category		
Size (FP)	Count	FP/PM (Median)
<=50	269	3.49
51-100	492	5.13
101-150	304	6.54
151-200	216	6.67
201-250	160	7.65
251-300	159	8.49
301-400	171	9.55
401-500	102	9.72
501-1000	204	13.43
1001-2000	97	16.29
>2000	57	23.10

Figure 1: Productivity as an Input Using an Assumed Factor

For 350 FP, the median productivity is 9.55 function points per person-month (FP/PM). Assuming 152 working hours per month[2], this corresponds to hourly productivity of 9.55/152 = 0.0628 FP/hour. Effort, in hours, is then estimated by calculating 350 divided by 0.0628.

- Analogy. In this method, the estimator obtains productivity from an analogous software project. For example, the ISBSG database[3] offers a way of filtering a large database of software projects, based on the selection of characteristics that are most analogous. The database can be filtered based on industry, year of project, application group, development type, and many other variables. Once the most analogous project is found, the estimator can look up that project's productivity and use it in the estimate[4].
- Database average, median, or midpoint. In this method, the estimator uses a database of many projects and calculates productivity based on a measure of central tendency, such as the average (arithmetic mean). For example, the

[2] This is the Constructive Cost Model (COCOMO) standard.
[3] International Software Benchmarking Standards Group, https://www.isbsg.org
[4] ICEAA CEBoK-S, Lesson 4: Estimating the Cost of Custom Software Development, slides 52-61

ISBSG database could be filtered to obtain multiple analogous projects, which the analyst uses to calculate an average productivity.

In each of the above methods, productivity is a direct input to the estimating equation.

Productivity as an Output

Productivity may also be viewed as an output. In these examples, an estimating equation is used but productivity is not a direct input. Instead, the estimator uses alternative ways of measuring software complexity and developer capability. For example:

- Effort Multiplier (EM) Factors in the Constructive Cost Model (COCOMO). In the COCOMO II model[5], a software project is rated using 15 attributes, known as EMs. The product of the EM ratings is used, along with software size and a measure of economy or diseconomy of scale/size to derive an effort estimate. Productivity can then be calculated *post-hoc* using the standard metric of effort divided by size.
- Custom Effort Estimating Relationship (EER). In this method the analyst starts with a database of analogous software projects that includes actual size and effort for each project. The analyst uses statistical techniques such as linear regression to derive a relationship between size and effort. The resulting equation may not contain an independent variable for productivity. However, productivity can still be calculated using size divided by effort.
- Effort Estimating Tool. There are several cost estimating tools that require the cost analyst to input software development characteristics. These tools then produce an effort estimate based on a combination of data and relationships. For example, the basic COCOMO estimating equation relates size to effort

[5] Constructive Cost Model, https://en.wikipedia.org/wiki/COCOMO

directly. In addition, the Technomics Software Effort Estimating Tool (SWEET)[6] uses ISBSG data to create a customized EER based on a limited set of inputs.

In each of these estimating methods, productivity is an inherent part of the approach. However, it is not directly input to the estimating equation. Therefore, productivity metrics must be obtained as an output, by dividing size by effort.

Summary of Productivity Measurement Methods

It is important to note that regardless of which estimating method is used, the concept of productivity is always present. Further, because productivity is never precisely known in advance, there is always associated risk and uncertainty. Therefore, it is important to consider not only how each method handles productivity, but also how each method handles risk and uncertainty. The following table summarizes this concept:

Table 1: Typical Treatment of Uncertainty by Productivity Estimating Method

Estimating Method	Treatment of Productivity	Typical Treatment of Uncertainty
Assumed Factor	Input, based on published rule of thumb	None or plus or minus 10%*
Analogy	Input, based on selection of an analogous project	None or plus or minus 10%*
Database Average	Input, based on an average of analogous projects	None, plus or minus 10%, or calculated standard deviation*
COCOMO	Output, based on EM factors	Not included in the basic equation*
Custom EER	Output, based on input variables	Calculated using standard error of the estimate, using an *assumed* distribution
EER Tool	Output, based on input variables	Available only if included in the tool

** Note: In each of the input-based methods, the analyst may be able to address uncertainty using a Monte Carlo analysis. This requires the analyst to have an uncertainty distribution for each input and access to a tool (e.g., Excel and Crystal Ball) that enables Monte Carlo analysis.*

[6] Dynamic Software Effort Estimation, Gellatly/Jones/Wekluk/Brown/Braxton, https://www.iceaaonline.com/wp-content/uploads/2022/06/SA07-Gellatly-Paper-Dynamic-Software-Effort-Estimation.pdf

Software Size and Effort

Because productivity is composed of size and effort, we first explore each of these variables, and address the risk and uncertainty in each.

Size

Size is a measure of the quantity of software developed. There are various ways an analyst may quantify and measure software size. The longest-standing method is source lines of code (SLOC). In a SLOC-based size estimate, the number of lines of delivered code are counted. Comment and blank lines are usually omitted. A variant of SLOC, known as equivalent source lines of code (ESLOC) adjusts the SLOC number down, based on the amount of reused, adapted, converted, auto-generated, and otherwise modified code[7].

While SLOC and ESLOC have a long history, alternative methods have more recently been devised. These address two fundamental problems with SLOC: (1) It is easy to count for delivered code, but hard to predict for planned projects; and (2) Modern software development uses visual tools and techniques that are not driven by SLOC.

A frequently used alternative to SLOC sizing is Function Point Analysis (FPA). This method is defined by the International Function Point Users Group (IFPUG)[8] and others. FPA benefits include estimation earlier in the software life cycle, direct connection to requirements, and technology and platform independence. The data and analysis in this paper use FPA as the sizing metric. This is consistent with modern software estimating techniques and offers the largest dataset that contains both size and effort data.

Because it is impossible to precisely know software size for a planned project, we need to address uncertainty within the size metric. To begin, we analyze the ISBSG database. This dataset, which is more thoroughly introduced in the following section,

[7] ICEAA CEBoK-S, Lesson X: Software Size, slides 11-27
[8] https://ifpug.org/

offers many projects that have reported size using IFPUG 4+ sizing[9] and AFP. We selected IFPUG 4+ AFP as our preferred sizing metric because (1) it is complexity-adjusted and (2) the selection provides the largest possible dataset among the functional sizing methods. The following table shows ISBSG summary statistics on IFPUG 4+ AFP projects:

Table 2: Descriptive Statistics of AFP-Denominated Size in the Dataset (n=1,675)

Statistic	Size
Mean	128.5
Median	87.0
StDev	126.1
Skewness	0.988
CV	98%
Min	7
Max	2,048

One thing to note from this data is the high coefficient of variation (CV) value. The CV of approximately 100% indicates a high level of variability in the data: the standard deviation and mean are approximately equal. Additionally, the distribution of size is right-skewed. This is also evident in Table 2, which shows a skewness value of 0.988 (positive skewness values indicate right-skew).

Effort

The second component of productivity is effort. This metric is simply the number of man-hours required for software development. In our data, we use effort that includes work to design, build, test, and implement the software, because these are the “core” activities in ISBSG—and the activities for which the dataset is most complete. We exclude hours reported for other activities such as planning and specification.

Using the same ISBSG dataset, we find the following summary statistics on effort:

[9] IFPUG 4+ sizing refers to the version of the published Function Point Counting Practices Manual. The current version is 4.3.1.

Table 3: Descriptive Statistics of Effort Hours in the Dataset (n=1,675)

Statistic	Effort
Mean	1,744
Median	1,165
StDev	2,103
Skewness	0.826
CV	121%
Min	0
Max	35,063

Consistent with the size data, we see a high CV value, indicating a high level of variability in the data. A combined graph showing the frequencies of size and effort (akin to Probability Density Function (PDFs)) is shown in the figure below.

Figure 2: Frequencies of Size and Effort by Bin Percentile

The graph uses the concept of *binned* data, which is discussed in greater detail later in this paper. To create the graph, we separated the size and effort data into 53 evenly spaced intervals, or “bins.” The lowest size bin contains the lowest 1/53rd of the size data, and ranges from 0 to 38.6 AFP. Similarly, the lowest effort bin contains the lowest 1/53rd of the effort data, and ranges from 0 to 661.6 effort hours. The bin percentile of this lowest bin, in both cases, is 1/53, expressed as a percentage (about 1.89%). The graph plots the frequencies (counts) of size and effort data by bin percentile, for each of the 53 bins.

The graph shows that the distributions of size and effort are positively correlated and are each right-skewed. This result is not surprising. We expect size and effort to be highly correlated. In other words, large projects require a large amount of effort, whereas small projects require a small amount of effort. Because size and effort are positively correlated, it is not immediately obvious what the distribution around productivity would be. For example, if size and effort were perfectly correlated and proportional, then we would see only a single productivity number with no uncertainty. On the other hand, productivity as a random variable combines variance from two other random variables. In situations like this, uncertainty can compound, rather than diminish. In general, there is no closed-form formula for the variance of a ratio. It can only be approximated[10].Therefore, both the variance and nature of the distribution around productivity are difficult to predict. We start our exploration of this topic in the next section, with the same summary statistics on productivity.

Initial Analysis of Productivity

Using our standard definition of productivity (size divided by effort), we begin by calculating productivity on our dataset by taking AFP size and dividing by effort man-hours. The results can also be reported using summary statistics:

Table 4: Descriptive Statistics of Productivity in the Dataset

Statistic	Productivity (AFP/MH)
Mean	0.1369
Median	0.0876
StDev	0.1601
Skewness	0.923
CV	117%
Min	0.0031
Max	2.0667

[10] https://www.stat.cmu.edu/%7Ehseltman/files/ratio.pdf. The approximation can be written as: $\mathrm{Var}(S/E) \approx (\mu_s/\mu_e)^2\,[(\mu_s/\sigma_s)^2 + (\mu_e/\sigma_e)^2 - 2\mathrm{Cov}(S,E)/(\mu_s\mu_e)]$, where S represents size and E represents effort. This form makes clear the positive correlation between the covariates lowers the variance of their quotient, but their contributions are otherwise *additive*.

Interestingly, the CV of 117% indicates more variance than seen in the size data. Although less than the effort data CV, it again indicates a high level of variability in productivity. The following sections further explore this concept. We will use analysis to answer questions such as:

- How much variance is present in productivity data?
- What probability distribution should be used to model productivity?
- What distribution parameters best fit our data?

We start by looking deeper into the distribution of productivity data and discussing expected results.

Expected Distribution of Productivity

Due to the uncertainty associated with size and effort addressed in the previous section, we expect productivity to also contain a high level of uncertainty. Therefore, this necessarily leads to an important conclusion:

Software development productivity should never be treated as a known variable. It must always contain a measure of uncertainty.

In the following sections, we further explore our productivity data.

Background

We have defined Productivity[11] as a stochastic variable, expressed as a fraction, quotient, or ratio. The numerator and denominator (Size and Effort, respectively) of the fraction are themselves stochastic. This setting describes a Ratio Distribution[12]. If X and Y are random variables, then the Z variable follows a Ratio Distribution.

[11] Capitalization of terms such as “Size,” “Effort,” and “Productivity” denotes random variables with those respective names. For example, we model software size by with a Size variable that follows a certain distribution.
[12] https://mathworld.wolfram.com/RatioDistribution.html

$$Z = X/Y$$

There is a significant body of literature on Ratio Distributions. Specifically, if X and Y are independent, zero-mean Normal random variables, then Z follows a Cauchy distribution[13], which is difficult to characterize. In fact, the Cauchy distribution has no defined mean or variance, and is sometimes called a *pathological* distribution[14]. Under certain assumptions, Cauchy distributions can be approximated in Microsoft Excel.

However, in this context, there is reason to believe that Size and Effort are each right-skewed (non-normal), and positively correlated with each other:

Table 5: Descriptive Statistics for Size, Effort, and Productivity

Statistic	Size	Effort	Productivity
Mean	128.5	1,744	0.1369
Median	87.0	1,165	0.0876
StDev	126.1	2,103	0.1601
Skewness	0.988	0.826	0.923
CV	98%	121%	117%
Min	7	48.54	0.0031
Max	2,048	35,063	2.0667

[13] https://en.wikipedia.org/wiki/Cauchy_distribution
[14] http://www.math.nagoya-u.ac.jp/~richard/teaching/s2019/Cauchy_Distribution.pdf

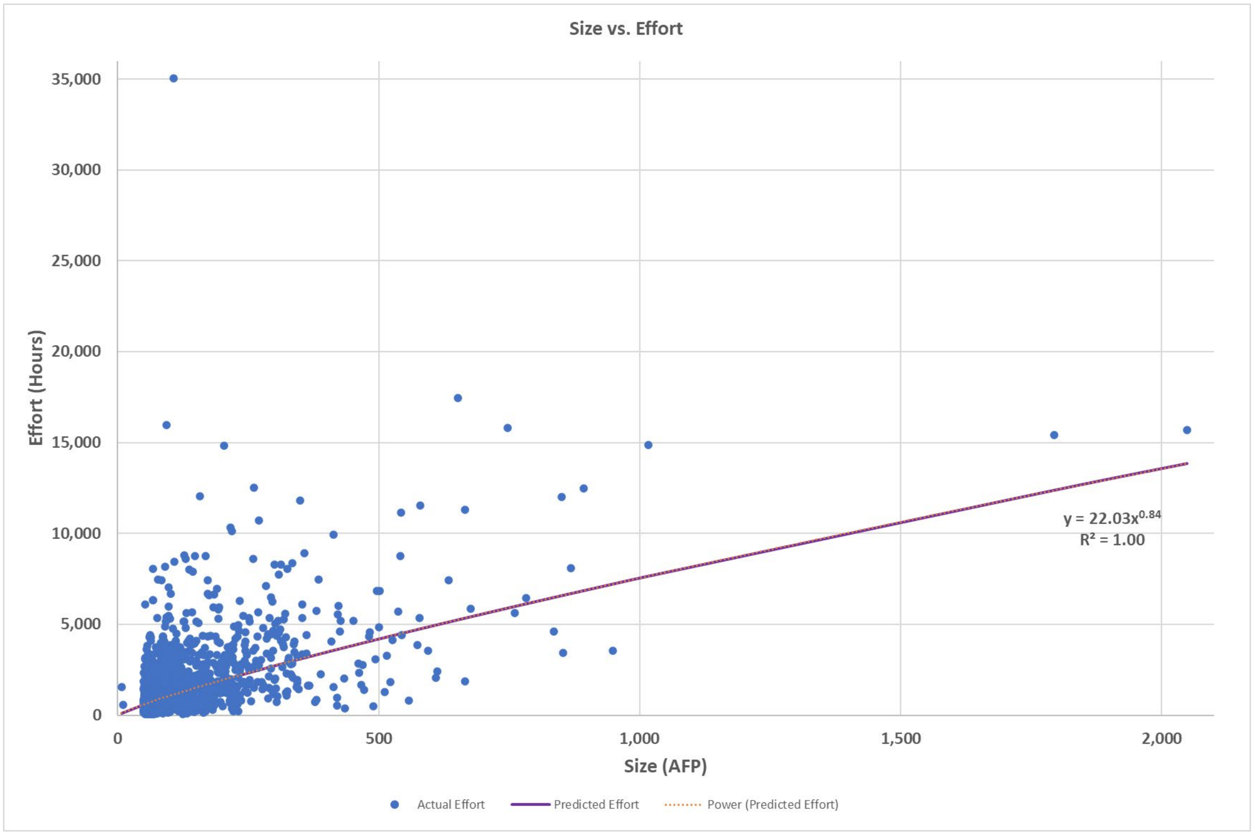


Figure 3: Scatterplot of Size (AFP, x-axis) vs. Effort (Hours, y-axis)

In Table 5 above, there is a strong positive skewness for both Size and Effort. Moreover, Figure 3 demonstrates a positive correlation between Size and Effort: the correlation is 100.00% (to two decimals). Therefore, our setting is more complex than the Cauchy setting.

It is easy to think of real-world applications of other Ratio Distributions like ours. For example, the Price to Earnings (P/E) ratio of a stock is the quotient of two non-normal, right-skewed[15], positively correlated variables. Similarly, cost analysts often are concerned with Return on Investment (ROI) for specific initiatives, where Return and

[15] Right-skewness of stock prices and earnings follows from their general upward trend, and the fact that both are generally bounded at the low end, but not the high end. The fact that the two are positively correlated is well-established.

Investment are likely right-skewed and positively correlated with each other[16]. At least one analyst has observed that P/E ratios appear to follow a lognormal distribution[17].

Beyond the lognormal, which distributions are good candidates to approximate the ratio of size and effort?

Candidate Distributions

Although size and effort are each right-skewed, it's unclear initially that productivity should be right-skewed. Cost risk analysis consistently tells us that unfavorable (higher cost) outcomes are more likely than favorable ones. Thus, size and effort should be right-skewed, as we saw earlier.

However, higher productivity values are associated with *favorable* outcomes. Extreme events associated with *unfavorable* outcomes should be associated with lower Productivity, and therefore, *left* skew. The **Beta** distribution can accommodate almost any shape or skewness. Its versatility in this regard makes it a good candidate to attempt to "fit" to our productivity data. Cost analysts have proposed Beta distributions previously in the context of S-curves and time-phasing[18].

Cost analysts have also long proposed that **Weibull** distributions fit time-phased data better than traditional Earned Value Management (EVM)-type metrics. These analyses include fitting of Cumulative Distribution Functions (CDFs, or S-Curves) of both total cost and total duration. The seminal work in this regard is Peter Norden's (1963) discovery that software development costs' time-phasing generally follows a Rayleigh distribution. This work was so influential that Rayleigh distributions are often referred to,

[16] There is a similar logic here. In general, higher Investments lead to higher absolute Returns. Meanwhile, Investment cannot be negative, and Returns tend to compound over time, implying an asymmetric distribution.
[17] https://stats.stackexchange.com/questions/46141/what-is-the-distribution-that-can-properly-describe-the-pe-fluctuation-of-a-stoc
[18] Brown, White, Ritschel, and Seibel. *Time-Phasing Aircraft R&D Using Beta and Weibull Distributions*. Journal of Cost Analysis and Parametrics (2020.) https://www.iceaaonline.com/jcap2015-1096219/.

even today, as Norden-Rayleigh[19] distributions or curves[20]. Dr. David Lee's Rayleigh Analyzer[21] subsequently brought this idea into the cost analysis mainstream. Weibull distributions are generalizations of Rayleigh distributions; cost analysts continue to promote their use. In fact, we found at least seven ICEAA papers or training sessions studying the use of Weibull distributions to characterize cost and schedule data. One such session even referred to the Weibull distribution as "Gumby," presumably because it can bend or contort to fit almost any shape[22].

In any study of CDFs or S-Curves, we must include the two typical S-Curve distributions: **Normal** and **Lognormal**. In fact, these are the only S-Curve distributions one can select in the Naval Center for Cost Analysis (NCCA) S-Curve tool[23].

Finally, cost analysts often rely on Expert Opinion to estimate uncertainty parameters for cost-driving input variables. The most common of these characterizations is called Three Point Estimating, wherein a Subject Matter Expert (SME) specifies a Low, Most Likely, and High value for the parameter. Those three points then become the basis of a Program Evaluation Review Technique (PERT) Beta, or more commonly, a Triangular distribution. Because PERT Beta distributions are approximations of the Beta

[19] The discovery of Rayleigh distributions is attributed to the true forefather of this work, Lord Rayleigh (1877), predating both the discovery of the Weibull generalization (1933) and Norden's work (1963).

[20] See, for example, Jones, Alan. *Norden-Rayleigh Curves for Solution Development*. https://www.taylorfrancis.com/chapters/mono/10.4324/9781315160030-2/norden-rayleigh-curves-solution-development-alan-jones

[21] Lee, Dr. David A. *The Rayleigh Analyzer*. Logistics Management Institute (LMI) (1999). https://apps.dtic.mil/sti/tr/pdf/ADA371359.pdf.

[22] Stouffer (LMI, 2009) (using Weibull distributions to model failure rates); Rudolph (Technomics, 2010), touting the benefits of a "Weibullnator" tool to forecast ship construction costs; Garcia (MCR, 2013), using Weibull distributions to time-phase ground-based radar costs; Burgess, Smirnoff, and Wong (NRO, 2014), using Weibull distributions to time-phase space launch system costs; Lee, DeZwarte, Sigalas-Markham, and Eckhause (LMI, 2014), rejecting Weibull distributions in favor of "parsimonious" distributions; Hawpe and Mender, who referred to Weibull-related efforts as *Modeling with Gumby*, finding that they fit the data well but have little predictive power (NCCA, 2017); and the aforementioned paper by Brown et al (2020). All of these papers are available at https://www.iceaaonline.com/archives/

[23] https://www.dau.edu/index.php/tools/s-curve-tool-risk-uncertainty-analysis

distributions we are already considering, we did not consider them separately. However, in accordance with this common practice, we considered modeling productivity with a **Triangular** distribution.

Choosing the Best-Fitting Distribution(S)

The following sections describe the methods we used to calculate the best fitting parameters for each distribution.

Methods of Specifying Distributional Parameters

Once we have selected a set of candidate distributions, we must specify a method for characterizing their parameters. After all, we cannot consider every *possible* Beta distribution, every *possible* Weibull distribution, etc. Broadly speaking, there are three ways to specify these parameters:

1. Maximum Likelihood Estimation (MLE),
2. Method of Moments (MoM), and
3. Fitting to a data set via minimization of a penalty function.

Maximum Likelihood Estimation (MLE): This method refers to creating the likelihood function associated with each distribution, then finding the parameters that maximize it. We did not use MLE in this analysis, because it was more efficient to fit the distributional parameters to the data directly than to create likelihood functions for each distribution. We are confident that use of MLE would not significantly change the results.

Method of Moments (MoM): This method refers to assuming a distribution and using the sample statistics of a data set to imply its parameters. For example, one could assume that productivity follows a normal distribution, whose mean and standard deviation are the mean and standard deviation of the sample data. As cost analysts often follow this convention, one of the distributions we tested is called “Normal (MoM).” This refers to the normal distribution implied by MoM estimators.

Fitting via a Penalty Function: This method refers to using the best-fitting distributional parameters based on minimization of a “penalty function”. This is similar to the manner in which the Ordinary Least Squares (OLS) regression routine finds the best-fitting slope and y-intercept of a line based on minimization of Sum of Squared Errors (SSE). This is the primary method that we used—we applied it to each distribution we tested. The specifics of the distribution-fitting routine are discussed in the next subsection.

Stratification and Binning of Data

As previously discussed, we stratified the data based on size. Within each stratum, it is necessary to further "bin" the data. Why? Productivity is a continuous random variable that could take on infinitely many values. Therefore, the probability that Productivity will take on any *particular* value is zero. For continuous random variables, we can refer to probability, likelihood, or frequency, only for *ranges* of values. But how do we create these ranges, or bins? Is there an optimal "width" of each bin?

Many rules for binning data have been proposed. Some of them include:

1. Scott's Rule[24]
2. Sturge's Rule[25]
3. Doane's Rule[26]
4. Rice's Rule[27]
5. Freedman and Diaconis' Rule[28]

Each of these rules have specific assumptions, advantages, and limitations. A complete discussion of the pros and cons of each rule is beyond the scope of this paper. For simplicity, and because it is prominently featured in the Cost Estimating Body of

[24] Scott, D.W. (1992) Multivariate density estimation: theory, practice, and visualization, John Wiley & Sons: New York.
[25] Sturges, H. (1926) The choice of a class-interval. J. Amer. Statist. Assoc., 21, 65–66.
[26] Doane, D.P (1976). *Aesthetic frequency classification*. American Statistician, 30, 181– 183. Retrieved February 17, 2024 from http://www.jstor.org/stable/2683757
[27] Lane, D. M. (n.d.). Online Statistics Education: An Interactive Multimedia Course of Study. Rice University. Retrieved from http://onlinestatbook.com/
[28] Freedman, D. and Diaconis, P. (1981) On this histogram as a density estimator: L2 theory. Zeit. Wahr. ver. Geb., 57, 453–476

Knowledge (CEBoK), we chose Scott's Rule. This rule states that the optimal bin width is given by:

$$Bin\ width = \frac{3.49s}{\sqrt[3]{n}}$$

Where *s* is the sample standard deviation and *n* is the sample size. This implies a certain number of bins, which is rounded up to the nearest whole number.

Selecting the Penalty Function

The industry-standard test to determine whether binned data fit a theoretical distribution is the Chi-Square test. That test involves generating the Chi-Square Statistic (CSS), then obtaining an associated p-value. In this context, higher p-values are *favorable* because they indicate that the hypothesized distribution cannot reasonably be rejected. Lower p-values are unfavorable because they indicate rejection of the hypothesized distribution.

From this it follows that lower CSS values should be associated with higher p-values and a superior distribution fit. The formula for the CSS confirms this:

$$CSS = \sum_{i=1}^{j} \frac{(O_i - E_i)^2}{E_i}$$

where *O* and *E* are the observed and expected frequencies of data in each of the *j* bins. In this formula, if the distribution fits perfectly, then *O*=*E* for each bin, *CSS* = 0 and the associated p-value = 1.0000. At the other extreme, when the distribution fits very poorly, the CSS approaches infinity and the p-value approaches zero.

One limitation of the Chi-Square routine is that the expected frequency (*E*) for each bin must be at least five. As we will see, the distribution of actual productivity is so heavily right-skewed that no reasonable theoretical distribution will have this property. In addition, many of the distributions tested give similar results.

To address these issues, we used a standard SSE-based penalty function, but also applied the CSS to eligible bins[29]. For the large data set, the limitations of Excel Solver required focusing on only the first 11 bins[30]. Finally, we created "micro bins," for visualization only, to better show the differences among the distributions. All of the frequency-based graphs shown in this paper use micro bins.

Summary of the Method

The following tables summarize the method that we used, with associated rationale.

Table 6: List of Distributions Tested with Associated Rationales

Distribution	Method of Parameter Specification	Color Style in Figures	Rationale	Examples of Prior Work Using This Distribution
(Actual data)	Not Applicable	Light blue markers	This is the data set we are attempting to fit	Not Applicable
Beta	Minimize SSE	Red curves	Flexible, supports left skew, generalization of PERT Beta	Brown et al (2020)
Weibull	Minimize SSE	Light green curves	Flexible, "gumby distribution," generalization of Rayleigh curves	At least seven prior ICEAA papers or training sessions
Normal	Minimize SSE	Orange curves (solid)	Standard S-Curve distribution	NCCA S-Curve Tool
Lognormal	Minimize SSE	Purple curves	Standard S-Curve distribution	NCCA S-Curve Tool, DoD Joint Cost and Schedule Risk and Uncertainty Handbook[31]
Triangular	Minimize SSE	Black curves	Common cost analyst practice	Ubiquitous[32]
Normal (MoM)	MoM	Orange curves (dashed)	Common cost analyst practice	Ubiquitous

[29] This captured a very large percentage of the data.
[30] Again, this captured a very large percentage of the data—see Table 6
[31] https://mosaicprojects.com.au/PDF-Gen/CSRUH.pdf

[32] Most commonly, triangular distributions are specified based on Low, Most Likely, and High values provided by a SME (and perhaps adjusted for enhanced uncertainty), rather than fitted to data. However, we did not test a "Triangular (MoM)" distribution, as its fit was transparently extremely poor. Note that the fitted Triangular distribution could have "chosen" MoM parameters if those were the best fitting.

Table 7: Summary of Binning Conventions and Associated Purposes

Bin-Setting Method	Purpose	Number Of Bins (Small Projects)	Number Of Bins (Medium Projects)	Number Of Bins (Large Projects)
Scott's Rule	Minimize SSE[33]	21 of 21 (all bins)	17 of 17 (all bins)	11 of 27 (98.6% of data)
Scott's Rule	Chi-Square Test[34] (eligible bins)	6 of 21 (95.4% of data)	6 of 17 (91.4% of data)	6 of 27 (95.2% of data)
"Micro bins"	Visualization	47	50	37

Data Analysis

As previously stated, we used ISBSG data for this analysis. ISBSG was selected because it contains:

- A large volume of projects. ISBSG's software development repository contains 10,600 projects, dating from 1989 to 2020.
- A variety of descriptive variables (fields) on each project. A total of 252 fields are available. Of these, 105 are quantitative and 147 are qualitative.
- A variety of fields that describe software size. Although not all metrics are available for each project, ISBSG allows for the reporting of software size based on IFPUG function points, Common Software Measurement International Consortium (COSMIC) function points, Nesma function points, SLOC, and a variety of less prevalent metrics.
- Fields that identify development team work effort measured in person-hours.
- A data quality rating, which contains an ISBSG-assigned letter grade, A through D.

[33] This is the criterion we used to determine the best fitting distribution.

[34] This is the criterion we used to test the hypothesis that actual data follows each hypothesized distribution.

- Easy accessibility, through purchase of a subscription, to the general public. Details are available at https://www.isbsg.org/isbsg-subscriptions/.

It is important to note one limitation of the ISBSG dataset: not all the available fields are populated for all projects. For example, AFP are reported for approximately 71% of projects. As a result, the amount of data available for analysis is reduced, possibly significantly, depending on the type and number of fields chosen for analysis.

For this analysis, we needed to filter the ISBSG dataset. Prior to any filtering, ISBSG offers data that varies in data quality, type of software project, as well as the metrics reported for each project. We chose to filter the ISBSG according to the following criteria:

- Sizing metric. In order calculate and compare productivity over multiple projects, both size and effort must be measured the same way for every project. While effort is consistently measured in man-hours, the ISBSG data includes many ways of measuring size. The most prevalent method is using function point sizing according to IFPUG 4+ standards. Because this widely used sizing metric offers the largest dataset, we filtered for projects that report effort using this standard.
- Data quality. ISBSG rates data quality for each project on a letter scale ranging from A to D. The rating is based on ISBSG’s assessment of data credibility and completeness. We chose to only include projects rated A or B.
- Project activity scope. This field indicates what tasks were included in the project work effort data recorded. These are: (1) Planning, (2) Specify, (3) Design, (4) Build, (5) Test and (6) Implement. The most reported set of activities are “design; build; test; implement”. Because this scope of activities is a frequently seen, we chose to filter for projects that reported this scope. It is critical to note that reader use of our analysis requires a good understanding of the project being estimated. That is, if the project being estimated includes activities that differ, then the productivity analysis and calculations in this paper should not be used.

After filtering the ISBSG dataset, 1,675 records were available for analysis. Each record represents a software development project.

The next step was to split this curated ISBSG dataset, based on software size. Three project size categories - small, medium, and large projects – were created by placing one third of the data into each category. The reason for categorizing the data in this way is that it allows us to address and analyze differences in productivity due to economy or diseconomy of scale. For example, if productivity has different values or different uncertainty for small projects, then this approach will reveal that result.

A second reason for categorizing results by size is that it allows analysts using our results to select a category that most closely resembles the project of interest. This is particularly important for estimators working on large, U.S. federal government projects . We make this distinction because the ISBSG dataset is populated with many private industry projects that tend to be much smaller than a typical U.S government acquisition. The size categories enable estimators to choose projects whose size is most analogous to the project being estimated.

The final step to create the curated dataset was to calculate productivity based on size (measured in adjusted IFPUG 4+ function points) divided by effort (measured in development team man-hours).

Small Projects

Starting with the small category, we have 567 data points that are characterized by the following statistics:

Table 8: Small Project Statistics (n=567)

Statistic	Size	Effort	Productivity
Mean	59.4	936	0.1381
Median	59.0	716	0.0826
StDev	6.9	888	0.1667
Skewness	0.181	0.742	0.999
CV	12%	95%	121%
Min	7	48.54	0.0045
Max	70	8,044	1.4421

The small category is populated with projects that are less than or equal to 70 AFP. The distribution of productivity within this category is plotted using a histogram. This required binning the data. The number of bins and bin width was calculated using Scott's Bin Width, as described previously. For this category, Scott's bin width was calculated to be 0.070. To force each graph to begin at the origin, all minima were treated as zero for purposes of bin width calculations. Hence, taking the range of productivity values (0.0000 to 1.4421) and dividing by the bin width suggests that the optimal number of bins is 20.513, which was rounded up to 21 bins. Dividing the data into 21 equally sized bins results in an adjusted bin width of 0.0687, meaning that the first bar on the histogram would include productivity values between 0 and 0.0687. The second bar will include values from 0.0687 to 0.1373, and so on. The resulting histogram is shown below:

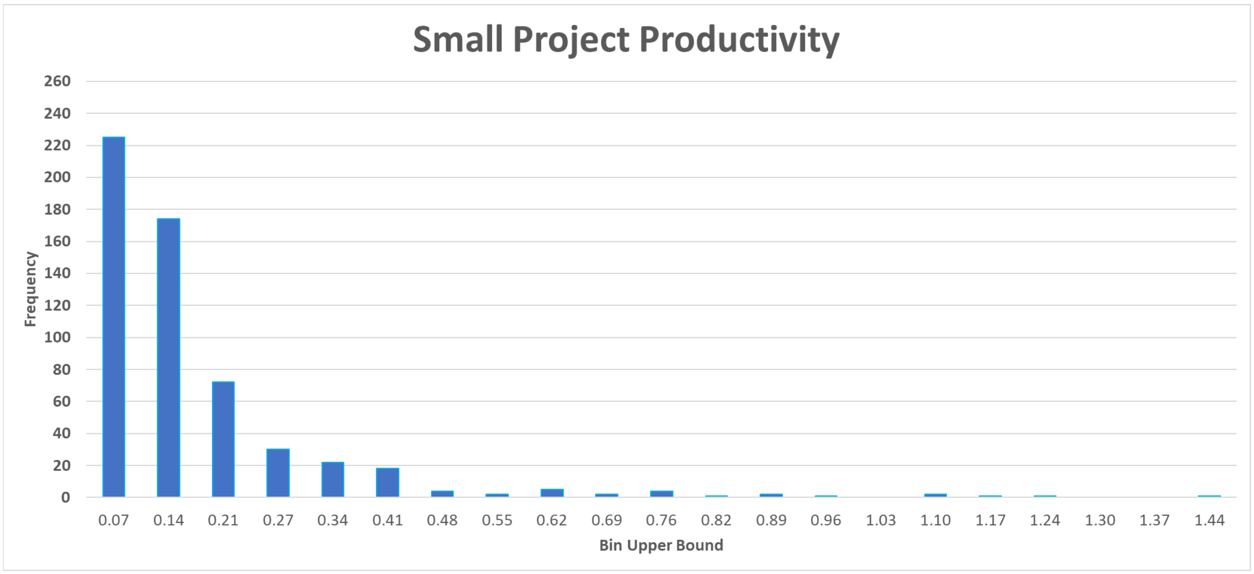


Figure 4: Small Project Productivity Histogram (n=567)

Visually, the distribution is heavily right skewed due to the small number of projects at the far-right end of the graph. This observation is confirmed by the fact that the mean is greater than the median[35]. As shown in Table 8, the median productivity for small projects is 0.0826 AFP/hour, which lies in the 2nd bin from the left. The mean is 0.1381 AFP/hour, which lies in the 3rd bin from the left and is clearly well beyond the 50th percentile. The presence of skewness indicates that a normal distribution may not be the best fit for the observed data.

The next step was to evaluate a variety of probability distributions. For this analysis, we evaluated the following distributions:

- Beta
- Weibull
- Normal
- Lognormal
- Triangular
- Normal (Method of Moments)

The rationale for each distribution's inclusion was described previously. For each distribution, we calculated the parameters that best fit the observed data. For example, the normal distribution is defined by the mean and standard deviation. We chose values for these parameters that allowed the normal distribution to best fit our data. This was accomplished by minimizing the sum of squared errors (SSE) between the observed and predicted values. Excel Solver was used to find the parameters for each distribution that minimized SSE. The following table shows the calculated results:

[35] Skewness is calculated by 3 * (mean – median) / Standard deviation. When the value is greater than zero (positive skewness), the curve is right-skewed. Negative skewness indicates a left-skewed curve.

Table 9: Fitted Distributions for Small Projects

BIN	LB	UB	CUMFREQ	FREQ	PRED_BETA	PRED_WEIBULL	PRED_NORMAL	PRED_LOGNORMAL	TRIANGULAR_CUM	PRED_TRIANGULAR	PRED_NORMAL_MoM
0	0	0	0	0	0	0	0	0	0	0	0.0
1	0.0000	0.0687	225	225	232.6	233.0	242.4	227.2	0.4256	241.3	192.0
2	0.0687	0.1373	399	174	172.8	170.2	177.8	173.3	0.7430	180.0	90.5
3	0.1373	0.2060	471	72	90.4	90.5	107.1	77.6	0.9345	108.6	90.7
4	0.2060	0.2747	501	30	42.1	42.6	33.7	37.6	1.0000	37.1	76.8
5	0.2747	0.3434	523	22	18.0	18.5	5.5	19.9	1.0000	0.0	55.1
6	0.3434	0.4120	541	18	7.1	7.6	0.5	11.2	1.0000	0.0	33.4
7	0.4120	0.4807	545	4	2.6	3.0	0.0	6.7	1.0000	0.0	17.1
8	0.4807	0.5494	547	2	0.9	1.1	0.0	4.2	1.0000	0.0	7.4
9	0.5494	0.6180	552	5	0.3	0.4	0.0	2.7	1.0000	0.0	2.7
10	0.6180	0.6867	554	2	0.1	0.1	0.0	1.8	1.0000	0.0	0.8
11	0.6867	0.7554	558	4	0.0	0.0	0.0	1.3	1.0000	0.0	0.2
12	0.7554	0.8241	559	1	0.0	0.0	0.0	0.9	1.0000	0.0	0.0
13	0.8241	0.8927	561	2	0.0	0.0	0.0	0.6	1.0000	0.0	0.0
14	0.8927	0.9614	562	1	0.0	0.0	0.0	0.5	1.0000	0.0	0.0
15	0.9614	1.0301	562	0	0.0	0.0	0.0	0.3	1.0000	0.0	0.0
16	1.0301	1.0988	564	2	0.0	0.0	0.0	0.3	1.0000	0.0	0.0
17	1.0988	1.1674	565	1	0.0	0.0	0.0	0.2	1.0000	0.0	0.0
18	1.1674	1.2361	566	1	0.0	0.0	0.0	0.2	1.0000	0.0	0.0
19	1.2361	1.3048	566	0	0.0	0.0	0.0	0.1	1.0000	0.0	0.0
20	1.3048	1.3734	566	0	0.0	0.0	0.0	0.1	1.0000	0.0	0.0
21	1.3734	1.4421	567	1	0.0	0.0	0.0	0.1	1.0000	0.0	0.0
			Sums:	**567**	**567.0**	**567.0**	**567.0**	**566.7**		**567.0**	**567.0**
			SSE:		**739.4**	**752.3**	**2,222.7**	**176.5**		**2,574.6**	**12,170.7**
			p-value:		**0.0001**	**0.0004**	**0.0000**	**0.2829**		**--**	**0.0000**
			SSE Rank:		2	3	4	**1**		5	6
			p-value Rank:		3	2	5	**1**		--	4

The SSE values for each fitted distribution are highlighted yellow. The best fitting distribution is the one that minimizes the SSE value, which for this dataset is the Lognormal distribution.

We also calculated the significance of each fitted distribution using the Chi-Square Statistic (CSS), calculated bins with at least five projects[36]. The p-values, shown on the last row of the table, indicate the probability that we would observe data that varies from the theoretical distribution by at least as much as it does—assuming that the theoretical distribution is the right one. For example, the p-value of 0.2829 for Lognormal indicates a roughly 28% chance that, if the observed data were precisely Lognormal (with our fitted parameters), we would observe data like this. In this framework, the distribution

[36] Our original intent was to minimize the CSS directly. However, the CSS routine is valid only when the expected frequency of each bin (or at least, the great majority of bins) is at least five. The data are so heavily right-skewed that no reasonably fitting distribution will have this property. The numerator of each term in the CSS represents squared error, and those terms are summed. Therefore, we chose to minimize the simpler SSE expression. Nevertheless, we applied the CSS routine to bins where the *actual* frequency was at least five to calculate p-values. It is not possible to simply re-define the bins so that each one has at least five observations—that would violate Scott's Rule, as well as the basic principle that bins must be equally spaced.

with the highest p-value is the one that arguably best describes or “fits” the data, for CSS-eligible bins. By this standard, not only is Lognormal the best fitting distribution, but its p-value is over 700 times greater than the so-called “gumby” distribution, Weibull, which is second best. Meanwhile, the Lognormal’s SSE is at least 75% lower than that of the next-best-fitting distributions: Beta and Weibull.

The fitted parameters for each distribution are in the table below. For completeness, we also calculated MoM parameters for each candidate distribution. However, only the light-yellow highlighted distributional specifications are graphed and tested.

Table 10: Fitted and MoM Parameters for Small Projects

Parameter	Fitted	MoM
Beta Alpha	1.3693	0.4530
Beta Beta	17.1172	2.8281
Weibull Alpha	1.2307	0.0435
Weibull Beta	0.1152	0.4068
Normal Mean	0.0838	0.1381
Normal StDev	0.0828	0.1667
Lognormal Mean	-2.4588	-2.4078
Lognormal StDev	0.8729	0.8904
Triangular Low	0.0024	0.0045
Triangular Mode	0.0024	0.0343
Triangular High	0.2761	1.4421

The Fitted column shows the distribution parameters for each distribution. Additionally, we calculated parameters using a Method of Moments (MoM) approach. With a MoM approach, the analyst does not “fit” a distribution, but instead calculates statistical values from the dataset and uses those calculations to specify a distribution. For example, a MoM calculation for the normal distribution is simply the mean and standard deviation of the data. The Normal MoM parameters are particularly important because this is a method commonly used in the cost estimating community. This method is to assume that the data is normally distributed, calculate mean, calculate standard deviation, and then apply those values in a distribution, which is then fed into a Monte Carlo simulation. Because this method is widely used, it is worth exploring whether our best fitted result can improve on the technique.

These results can also be viewed by plotting the observed (actual) values and comparing them to each of the fitted curves. The following graph shows these results and adds a smoothed line to each fitted distribution. This graph, and the corresponding ones we created for medium and large-sized projects, use the “micro bins” concept described previously.

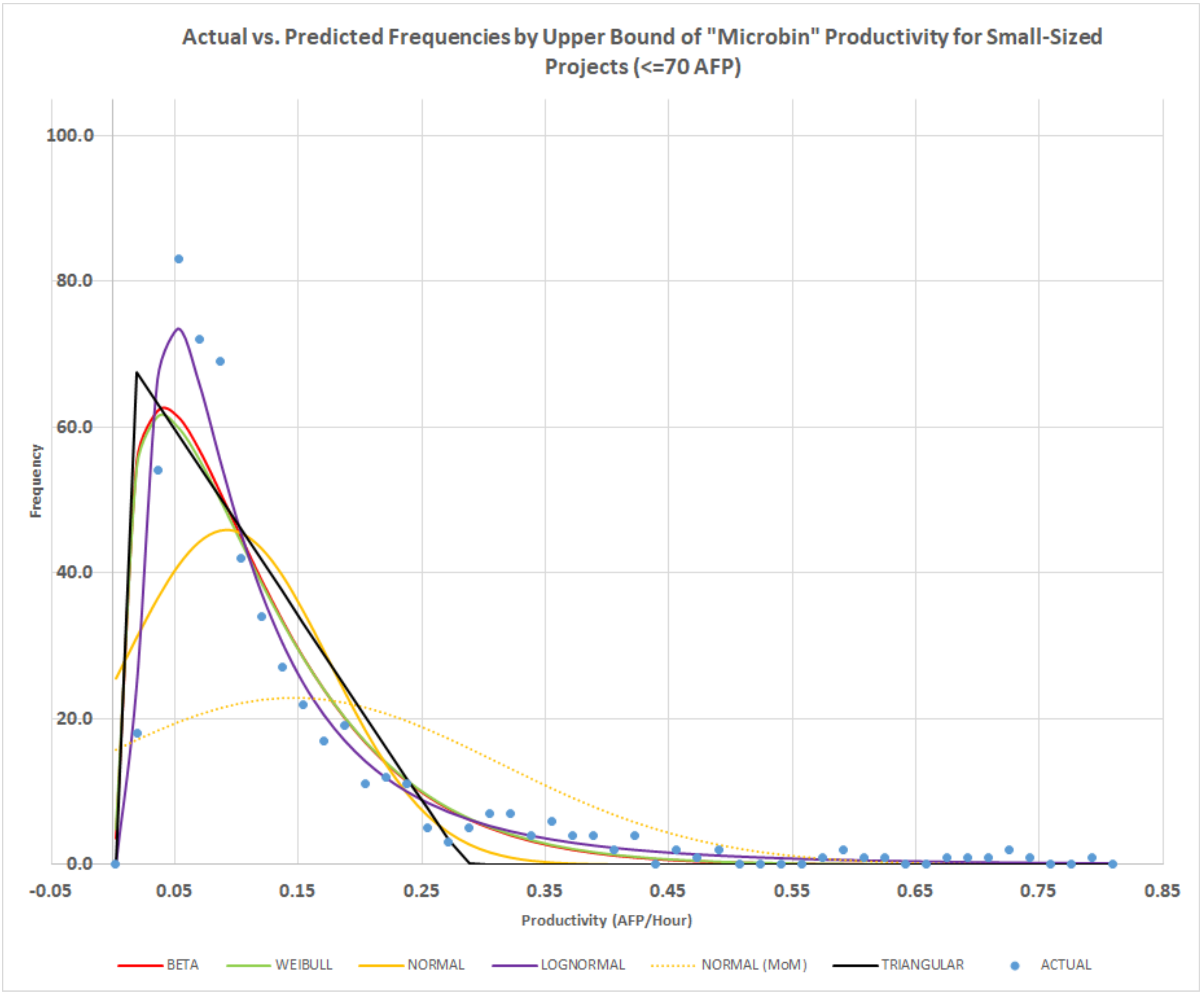


Figure 5: Comparison of Fitted Distributions for Small Projects

The graph (Figure 5) shows our actual data in the light blue dots. Although many of the lines overlap, we can see that at several points, the purple curve is clearly closer to the blue dots than the other lines. The Lognormal distribution was plotted using the purple curve, which confirms our quantitative conclusions that Lognormal is the best fitting curve.

This data can also be viewed as a cumulative distribution function (CDF), which is often referred to as an S-Curve. The same results, with cumulative percentile plotted on the y-axis are as follows:

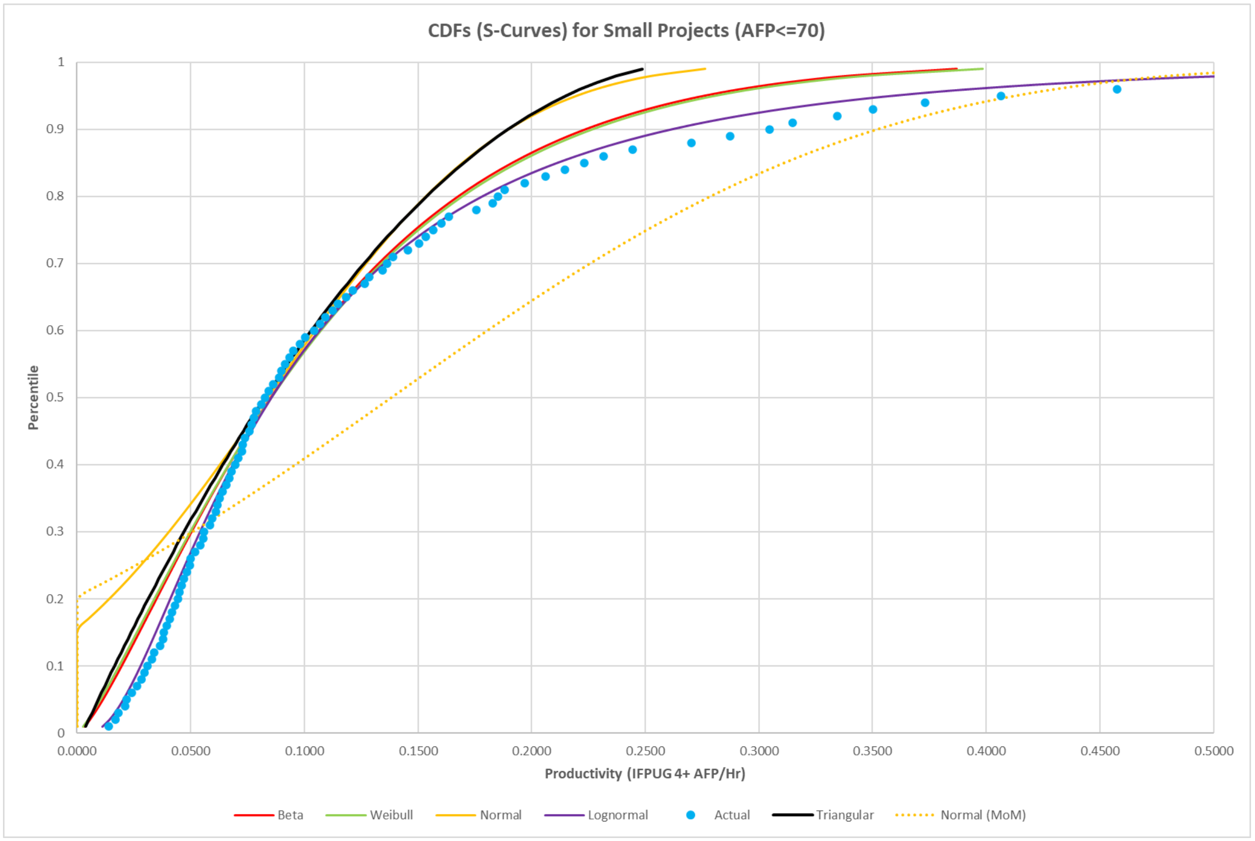


Figure 6: Comparison of Fitted CDF Distributions for Small Projects

These results indicate that for small projects (AFP less than 70), productivity can be best modeled using a Lognormal distribution, with parameters of Mean = -2.4588 and StDev = 0.8729[37].

[37] It is important not to interpret these numbers as direct productivities. The mean of a lognormal distribution is the mean of the natural logarithms (ln) of the underlying data. For example, if every productivity in the data set were 1 AFP/hour, the mean of the corresponding lognormal distribution would be zero, because ln(1) = 0. Here, the negative value for the distributional mean implies that productivity is generally less than 1 AFP/hour. This should be evident from Figure 6, where productivities of more than 0.5 AFP/hour are, in a literal sense, “off the chart.”

Medium Projects

Similar analysis was completed for medium sized projects. In our dataset, medium projects are greater than 70 AFP and less than or equal to 115 AFP. The summary statistics for this category are shown below:

Table 11: Medium Project Statistics (n=549)

Statistic	Size	Effort	Productivity
Mean	89.7	1,500	0.1203
Median	88.0	1,096	0.0805
StDev	13.0	1,970	0.1206
Skewness	0.382	0.615	0.989
CV	14%	131%	100%
Min	71	91	0.0031
Max	115	35,063	0.8242

The histogram of observed data for the medium category is as follows:

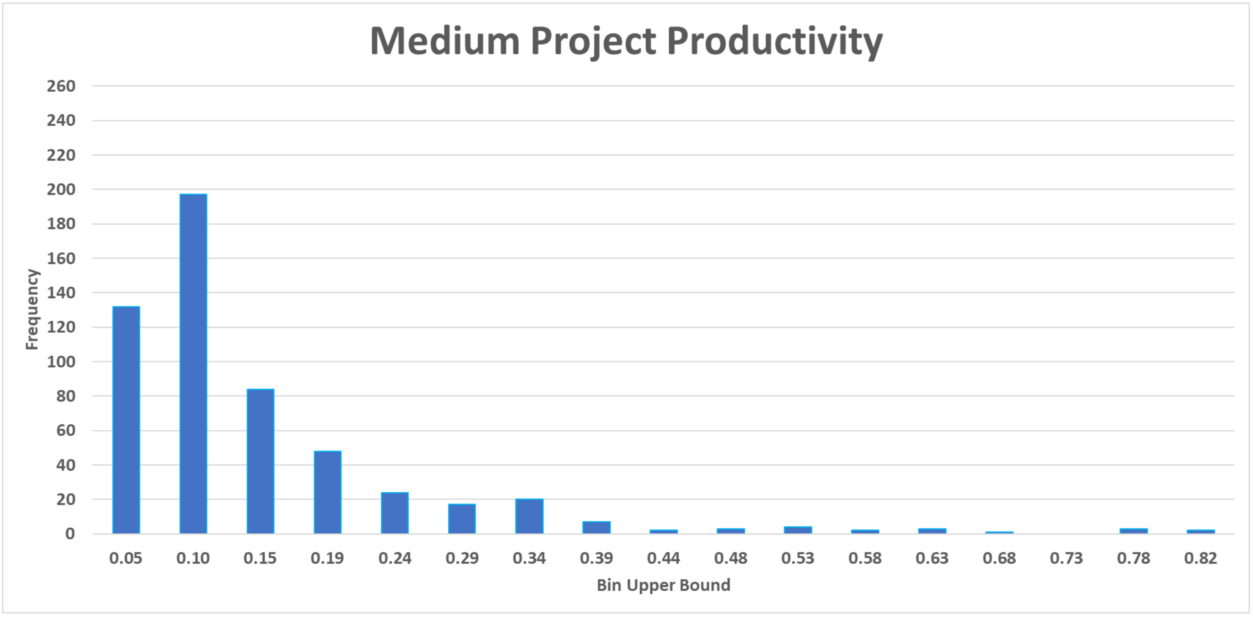


Figure 7: Medium Project Productivity Histogram (n=549)

Our analysis of the medium category paralleled that of the small category. We evaluated the same distributions and optimized each one using Excel Solver to select parameters that minimize SSE. The quantitative results are shown in the following table:

Table 12: Fitted Distributions for Medium Projects

BIN	LB	UB	CUMFREQ	FREQ	PRED_BETA	PRED_WEIBULL	PRED_NORMAL	PRED_LOGNORMAL	TRIANGULAR_CUM	PRED_TRIANGULAR	PRED_NORMAL_MoM
0	0	0	0	0	0	0	0	0	0	0	0
1	0.0000	0.0485	132	132	147.9	151.5	161.9	138.6	0.2665	146.3	151.4
2	0.0485	0.0970	329	197	190.7	184.4	195.6	192.1	0.6100	188.6	81.0
3	0.0970	0.1454	413	84	118.5	120.1	139.8	102.2	0.8459	129.5	87.5
4	0.1454	0.1939	461	48	57.1	58.8	44.8	51.4	0.9742	70.4	80.5
5	0.1939	0.2424	485	24	23.3	23.4	6.4	26.9	1.0000	14.2	63.2
6	0.2424	0.2909	502	17	8.2	7.9	0.4	14.8	1.0000	0.0	42.3
7	0.2909	0.3394	522	20	2.5	2.3	0.0	8.5	1.0000	0.0	24.1
8	0.3394	0.3878	529	7	0.6	0.6	0.0	5.1	1.0000	0.0	11.7
9	0.3878	0.4363	531	2	0.1	0.1	0.0	3.1	1.0000	0.0	4.9
10	0.4363	0.4848	534	3	0.0	0.0	0.0	2.0	1.0000	0.0	1.7
11	0.4848	0.5333	538	4	0.0	0.0	0.0	1.3	1.0000	0.0	0.5
12	0.5333	0.5818	540	2	0.0	0.0	0.0	0.9	1.0000	0.0	0.1
13	0.5818	0.6303	543	3	0.0	0.0	0.0	0.6	1.0000	0.0	0.0
14	0.6303	0.6787	544	1	0.0	0.0	0.0	0.4	1.0000	0.0	0.0
15	0.6787	0.7272	544	0	0.0	0.0	0.0	0.3	1.0000	0.0	0.0
16	0.7272	0.7757	547	3	0.0	0.0	0.0	0.2	1.0000	0.0	0.0
17	0.7757	0.8242	549	2	0.0	0.0	0.0	0.2	1.0000	0.0	0.0
			Sums:	549.0	549.0	549.0	549.0	548.5		549.0	549.0
			SSE:		**2,047.1**	**2,449.2**	**5,116.2**	**588.1**		**3,741.1**	**17,158.4**
			p-value:		**0.0003**	**0.0001**	**0.0000**	**0.4727**		**--**	**0.0000**
			SSE Rank:		2	3	5	**1**		4	6
			p-value Rank:		2	3	5	**1**		--	4

The SSE values for each fitted distribution are highlighted yellow. The best fitting distribution is the one that minimizes the SSE value, which for this dataset is again the Lognormal distribution. The fitted and MoM parameters for each distribution are in the following table:

Table 13: Fitted and MoM Parameters for Medium Projects

Parameter	Fitted	MoM
Beta Alpha	1.9845	0.7552
Beta Beta	16.0489	5.5243
Weibull Alpha	1.5516	0.0599
Weibull Beta	0.1005	0.4988
Normal Mean	0.0767	0.1203
Normal StDev	0.0523	0.1206
Lognormal Mean	-2.5275	-2.4684
Lognormal StDev	0.7483	0.8238
Triangular Low	0.0185	0.0031
Triangular Mode	0.0185	0.0242
Triangular High	0.2275	0.8242

These results again can be viewed by plotting the observed (actual) values and comparing them to each of the fitted curves. The following graph shows these results and adds a smoothed line to each fitted distribution:

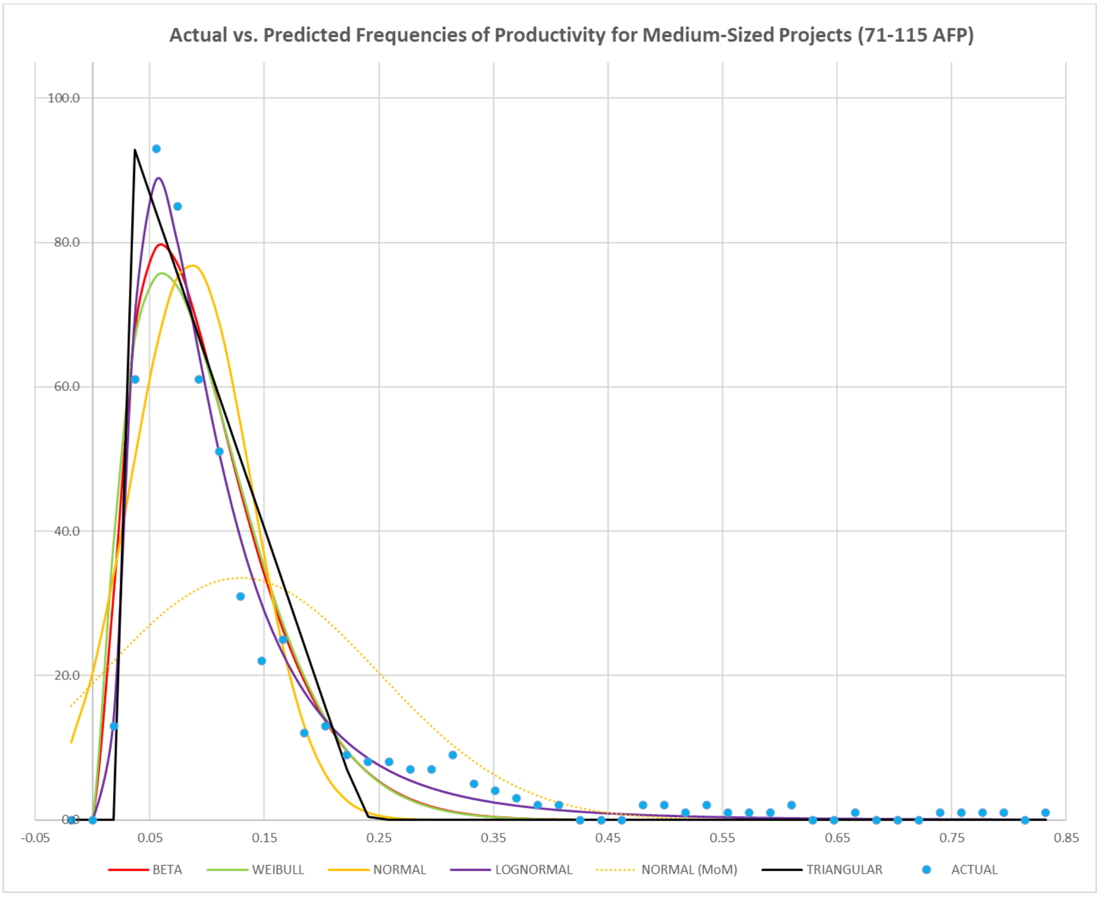


Figure 8: Comparison of Fitted Distributions for Medium Projects

The graph shows our actual data as light blue dots. Although many of the lines overlap, we can see that the purple curve is clearly closer to the blue dots than the other lines. The Lognormal distribution was plotted using the purple curve, which confirms our quantitative conclusions that Lognormal is the best fitting curve.

This data can also be viewed as a cumulative distribution function (CDF), which is often referred to as an S-Curve. The same results, with cumulative percentile plotted on the y-axis, are as follows:

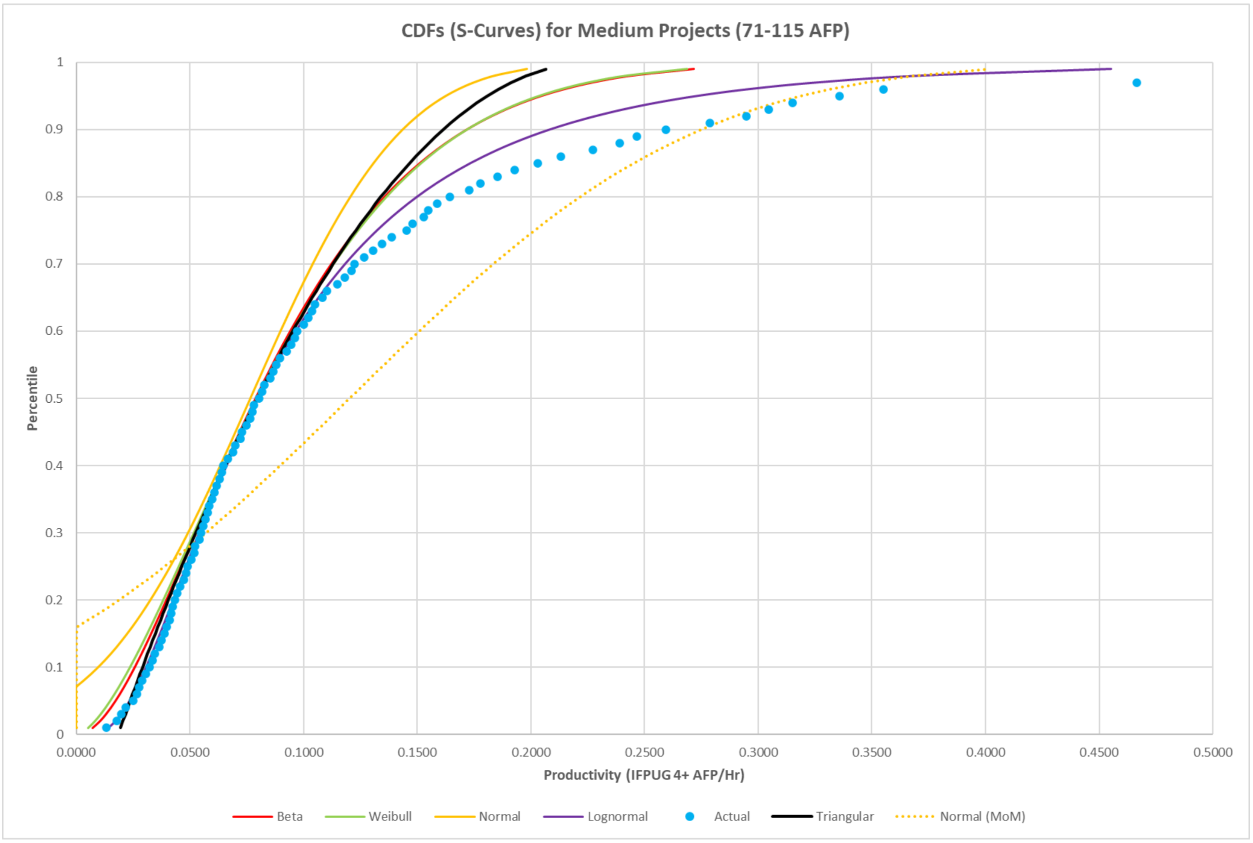


Figure 9: Comparison of Fitted CDF Distributions for Medium Projects

These results indicate that for medium projects (AFP 70 to 115) productivity can be best modelled using a Lognormal distribution, with parameters of Mean = -2.5275 and StDev = 0.7483.

Large Projects

Similar analysis was completed for large sized projects. In our dataset, large projects were those that are greater than 115 AFP. The summary statistics for this category are shown below:

Table 14: Large Project Statistics (n=549)

Statistic	Size	Effort	Productivity
Mean	236.8	2,804	0.1521
Median	185.0	1,882	0.0988
StDev	171.4	2,609	0.1842
Skewness	0.906	1.060	0.868
CV	72%	93%	121%
Min	116	60	0.0130
Max	2,048	17,444	2.0667

The histogram of observed data for the large category is as follows:

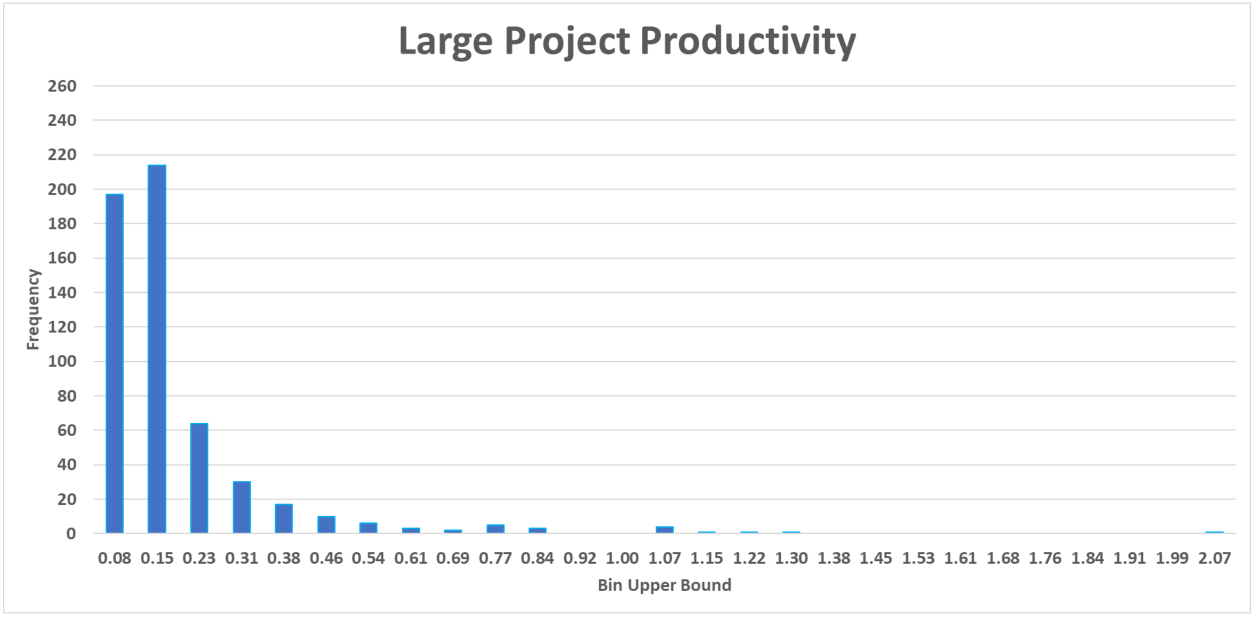


Figure 10: Large Project Productivity Histogram (n=559)

The same approach was taken in analyzing the large category. We evaluated the same distributions, and optimized each one, using Excel Solver, to select parameters that minimize SSE. The quantitative results are shown in the following table:

Table 15: Fitted Distributions for Large Projects

BIN	LB	UB	CUMFREQ	FREQ	PRED_BETA	PRED_WEIBULL	PRED_NORMAL	PRED_LOGNORMAL	TRIANGULAR_CUM	PRED_TRIANGULAR	PRED_NORMAL_MoM
0	0	0	0	0	0	0	0	0	0	0	0
1	0.0000	0.0765	197	197	209.4	212.2	218.1	203.7	0.3830	214.1	190.6
2	0.0765	0.1531	411	214	217.7	216.0	227.0	214.7	0.7729	218.0	90.2
3	0.1531	0.2296	475	64	92.6	96.3	99.1	83.0	0.9719	111.2	89.9
4	0.2296	0.3062	505	30	29.3	27.8	14.1	32.0	1.0000	15.7	75.7
5	0.3062	0.3827	522	17	7.8	5.7	0.6	13.3	1.0000	0.0	53.7
6	0.3827	0.4593	532	10	1.8	0.9	0.0	6.0	1.0000	0.0	32.2
7	0.4593	0.5358	538	6	0.4	0.1	0.0	2.9	1.0000	0.0	16.3
8	0.5358	0.6123	541	3	0.1	0.0	0.0	1.5	1.0000	0.0	6.9
9	0.6123	0.6889	543	2	0.0	0.0	0.0	0.8	1.0000	0.0	2.5
10	0.6889	0.7654	548	5	0.0	0.0	0.0	0.4	1.0000	0.0	0.8
11	0.7654	0.8420	551	3	0.0	0.0	0.0	0.3	1.0000	0.0	0.2
12	0.8420	0.9185	551	0	0.0	0.0	0.0	0.2	1.0000	0.0	0.0
13	0.9185	0.9951	551	0	0.0	0.0	0.0	0.1	1.0000	0.0	0.0
14	0.9951	1.0716	555	4	0.0	0.0	0.0	0.1	1.0000	0.0	0.0
15	1.0716	1.1481	556	1	0.0	0.0	0.0	0.0	1.0000	0.0	0.0
16	1.1481	1.2247	557	1	0.0	0.0	0.0	0.0	1.0000	0.0	0.0
17	1.2247	1.3012	558	1	0.0	0.0	0.0	0.0	1.0000	0.0	0.0
18	1.3012	1.3778	558	0	0.0	0.0	0.0	0.0	1.0000	0.0	0.0
19	1.3778	1.4543	558	0	0.0	0.0	0.0	0.0	1.0000	0.0	0.0
20	1.4543	1.5309	558	0	0.0	0.0	0.0	0.0	1.0000	0.0	0.0
21	1.5309	1.6074	558	0	0.0	0.0	0.0	0.0	1.0000	0.0	0.0
22	1.6074	1.6840	558	0	0.0	0.0	0.0	0.0	1.0000	0.0	0.0
23	1.6840	1.7605	558	0	0.0	0.0	0.0	0.0	1.0000	0.0	0.0
24	1.7605	1.8370	558	0	0.0	0.0	0.0	0.0	1.0000	0.0	0.0
25	1.8370	1.9136	558	0	0.0	0.0	0.0	0.0	1.0000	0.0	0.0
26	1.9136	1.9901	558	0	0.0	0.0	0.0	0.0	1.0000	0.0	0.0
27	1.9901	2.0667	559	1	0.0	0.0	0.0	0.0	1.0000	0.0	0.0
			Sums:	**559.0**	**559.0**	**559.0**	**559.0**	**559.0**		**559.0**	**559.0**
			SSE:		1236.4	1593.9	2568.4	500.2		3236.5	20150.4
			SSE (to first zero):		1216.4	1573.9	2548.4	480.8		3216.5	20130.4
			p-value:		0.0000	0.0000	0.0000	0.1381		--	0.0000

The SSE values for each fitted distribution are highlighted yellow. The best fitting distribution is the one that minimizes the SSE value[38], which for this dataset is again the Lognormal distribution. The optimized parameters for each distribution are in the following table:

[38] For the large dataset, we minimized SSE for only the non-zero bins, so that Excel solver would converge to a feasible solution. The second yellow highlighted row shows SSE for the non-zero bins.

Table 16: Optimized Parameters for Large Projects

	Fitted	**MoM**
Beta Alpha	2.2175	0.4255
Beta Beta	38.9826	2.3728
Weibull Alpha	1.6052	0.0474
Weibull Beta	0.1213	0.4052
Normal Mean	0.0958	0.1521
Normal StDev	0.0691	0.1842
Lognormal Mean	-2.3336	-2.4684
Lognormal StDev	0.6819	0.8238
Triangular Low	0.0234	0.0031
Triangular Mode	0.0234	0.0383
Triangular High	0.2712	2.0667

These results again be viewed by plotting the observed (actual) values and comparing them to each of the fitted curves. The following graph shows these results, and adds a smoothed line to each fitted distribution:

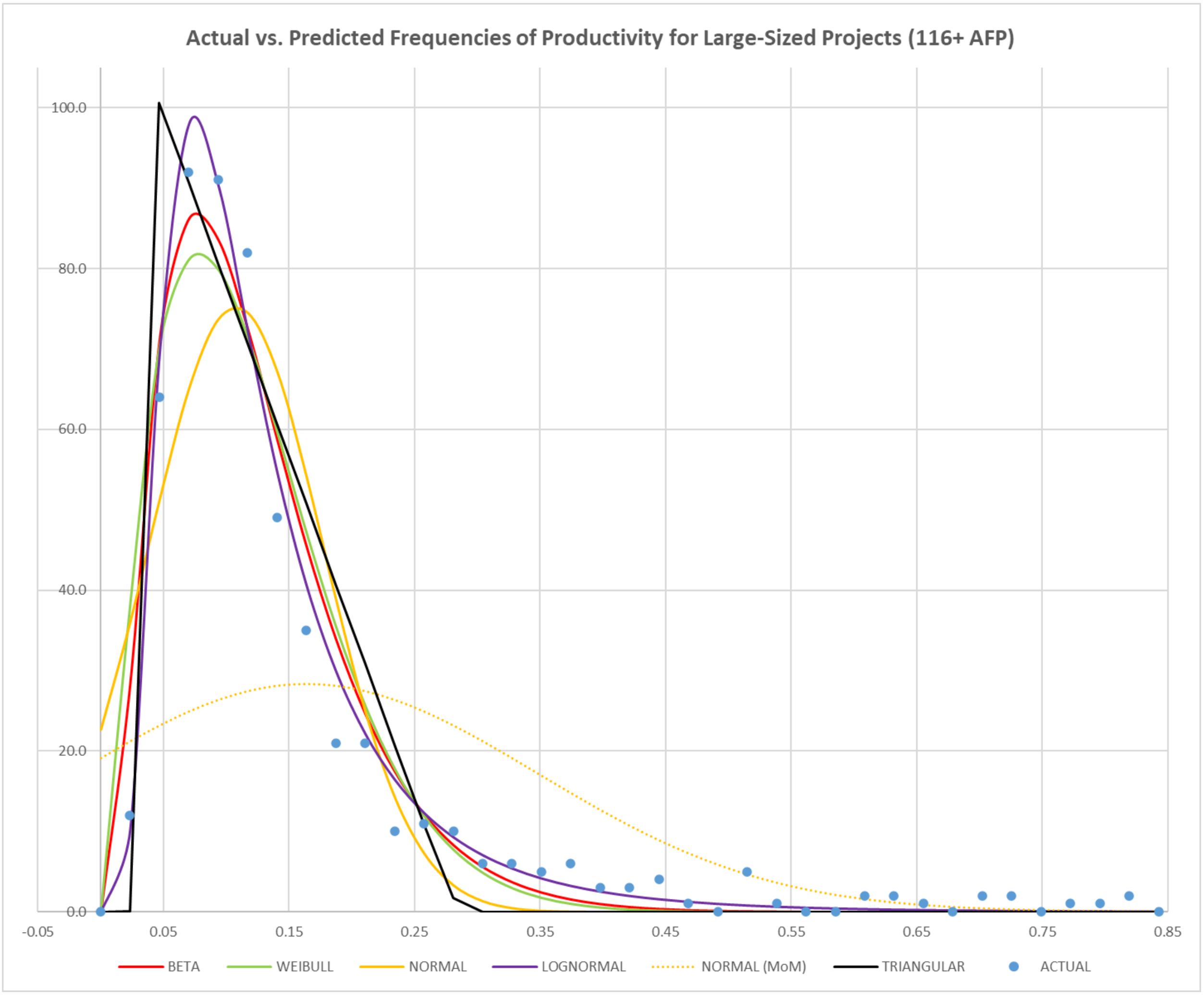


Figure 11: Comparison of Fitted Distributions for Large Projects

The graph shows our actual results as light blue dots. Although many of the lines overlap, we can see that in the middle of the graph, the purple line is clearly closer to the blue dots than the other lines. The Lognormal distribution was plotted using the purple line, which confirms our quantitative conclusions that Lognormal is the best fitting curve.

This data can also be viewed as a cumulative distribution function (CDF), which is often referred to as an S-Curve. The same results with cumulative percentile plotted on the Y axis are as follows:

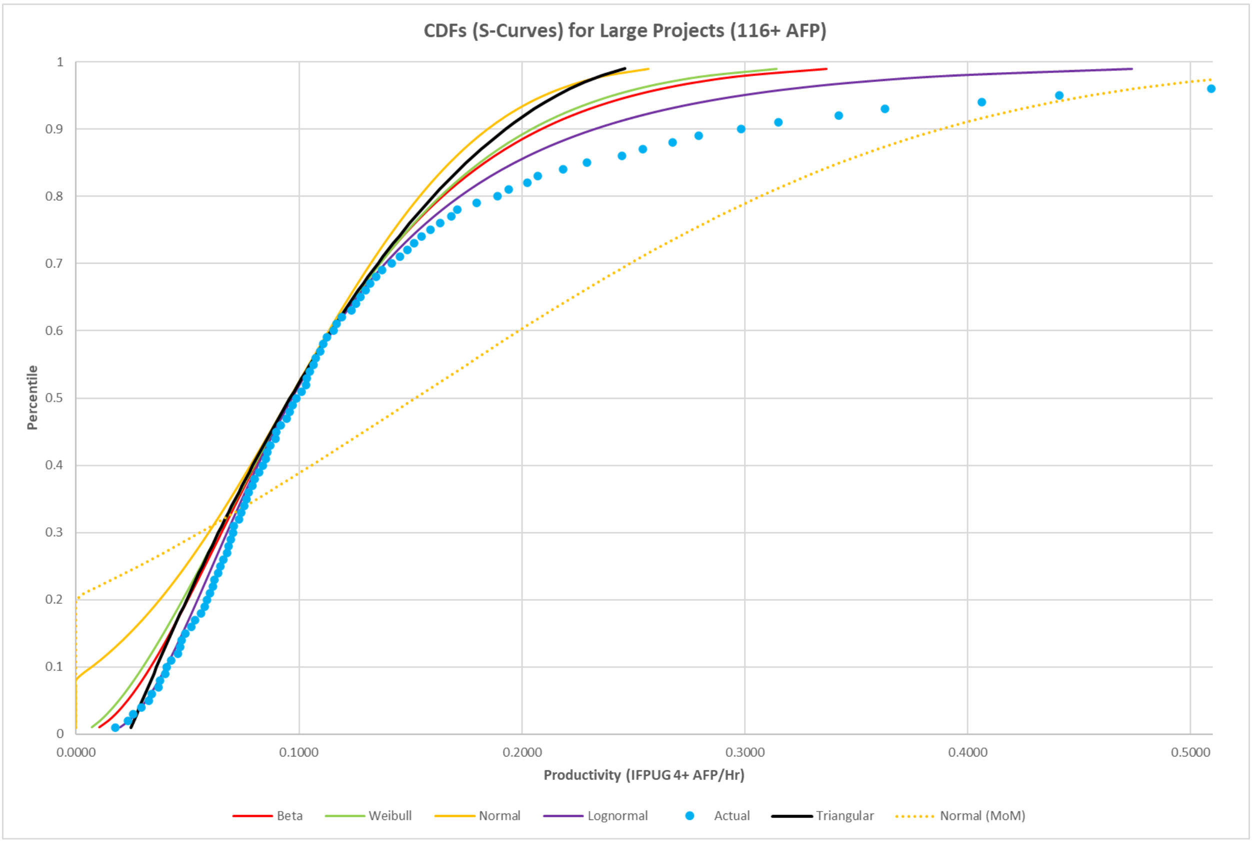


Figure 12: Comparison of Fitted CDF Distributions for Large Projects

Summary Results

The results from each size category are summarized in the table below:

Table 17: Summary Results for Each Size Category

Size Category	AFP Range	Number of Projects(n)	Best Fitting Curve	Curve Parameters
Small	0 to 70	567	Lognormal	Mean: -2.4558 StDev: 0.8729
Medium	71 to 115	549	Lognormal	Mean: -2.5275 StDev: 0.7483
Large	116+	559	Lognormal	Mean: -2.3336 StDev: 0.6819

In each dataset, the Lognormal distribution best fits the observed results.

Practical Applications

In this section, we explore how cost estimators should use our results to quantify productivity uncertainty. We offer several scenarios commonly seen in the software estimating community. For each scenario, we compare how the problem would typically be solved, and how that method could be improved with the findings of this paper.

Scenario 1: Estimate for an Agile Development Project

In our first example, we assume a software development estimate is needed for a project that employs an agile development methodology. Under this scenario, the estimator is given an effort metric. This is the reverse of a traditional software estimating problem where size is given. Under a strict agile philosophy, developers might be given a constrained schedule and team size. Therefore, effort is constrained and taken as an input.

Applying productivity and its uncertainty is particularly important under this scenario. Why? Because if effort is treated as a known and constrained variable, then ***all the risk and uncertainty*** will come from productivity. This productivity risk and uncertainty will translate into an estimated size metric, with its own risk and uncertainty. In other words, when an agile philosophy decides to constrain effort, then the project is inherently accepting scope risk, and the size of the product must be both estimated and risk-adjusted.

In this scenario, we assume the project effort is constrained to 1,100 man-hours. This value is near the median of our dataset, and therefore represents a realistic scenario. The estimator might then use the ISBSG dataset as follows:

- Filter ISBSG projects for data quality, use of IFPUG 4+ sizing, and development tasks. These are the same filter criteria we use in our analysis.
- From the filtered dataset, calculate productivity by dividing AFP by man-hours.
- Calculate a mean productivity. This results in mean = 0.1369
- Adjust for risk by varying the productivity number by plus and minus 10%. This method is one often employed by cost estimators who do not use Monte Carlo simulation.

Following these steps, the estimator would conclude that the calculated size is 151 AFP, with a range of 136 to 166 AFP.

Using the findings of this paper, we would follow this alternative process to the “traditional" method discussed above:

- Given our man-hour input is near the mean of our effort data, use our medium dataset.
- Apply a Lognormal distribution with mean = -2.5275 and StDev = 0.7483. These are the fitted parameters we show in Table 13.
- Calculate an 80% confidence interval by applying Excel’s LOGNORM.INV function, using 10% and 90% as the probability inputs.

The result is a range of productivity between 0.03 and 0.21. This results in a confidence interval for size of 34 to 229 AFP. The median AFP from this distribution is 88.

These are vastly different results. Our method yields a lower size estimate (88 versus 151), which indicates that less functional code will be delivered. But even more dramatic is that the risk range from our 80% confidence interval says that the AFP delivered could be as low as 34 AFP and as high as 229 AFP. The traditional method has vastly under-estimated the risk and uncertainty due to the productivity assumption.

Scenario 2: Estimate Effort for a Large Program

In our second example, we present a situation that is common to analysts estimating software development for a large government acquisition. The size of these types of projects is especially large compared to much of the data found in ISBSG.

For this example, we assume that software requirements have been documented, function point analysis has been done, and the size count is 1,500 AFP.

A typical estimating approach (i.e., one of the methods discussed earlier and shown in Figure 1) is to use the published productivity factors and assume that productivity is equal to 16.29 FP/PM. If we convert that to man-hours using 160 hours per month, the productivity factor is 0.1018. We then calculate effort based on size divided by productivity and get an effort estimate of 14,733 man-hours. As in the first scenario, a standard risk range of plus or minus 10% would give a range of 16,206 to 13,250 man-hours.

As an alternative, we use our large dataset, and again apply a Lognormal distribution with mean = -2.5275 and StDev = 0.7483. We get a median productivity value of 0.0970, with an 80% confidence interval of 0.0405 to 0.2323. Our effort estimate is therefore 15,472 man-hours, with a range of 6,457 to 37,075.

In this case, the two methods yield a similar point estimate of about 15K man-hours. But the risk range is vastly different. The high end of our range is 37K versus 16K using the traditional method. This suggests that the “worst-case” scenario” is far more effort than one would have assumed from the first calculation. If we apply a typical labor rate of $100 per hour, then the high end of the originally calculated range is $1.6M. When compared against our high end of $3.7M, it is easy to see how cost risk might be understated if productivity is not properly risk adjusted.

Scenario 3: Apply Uncertainty to a Known Productivity

Under this scenario, we assume that the analyst already has a credible estimate for productivity. For example, if productivity has been measured using analogous projects within the same agency or program, using historical actuals. In this case, the productivity value can be considered highly defensible. But the uncertainty and range around productivity still needs to be considered.

For this scenario, we assume that we are estimating the development effort for a large project that has been sized at 1,100 AFP. The defensible productivity for this project is 0.07 AFP/man-hour. Because the number is well founded, the analyst is treating it as a known value.

Using our results, we apply Excel’s `LOGNORM.DIST` function using the parameters we report in Table 16. The result is that the given productivity is found at the 32nd percentile of the lognormal CDF. The 10% and 90% percentiles are 0.0405 and 0.2323. We use these values to calculate an 80% confidence interval for effort. The result is an effort estimate of 16,224 man-hours, with a range of 4,735 to 27,188 man-hours. Using the same labor rate assumption of $100 per hour, this range translates to $473K to $2.7M.

This scenario demonstrates that even when an analyst has an acceptable number for productivity, our results can add context and value by providing a range and noting the confidence level of the given number. This can be accomplished quickly, without the need for Monte Carlo simulation.

The following two tables compare the results obtained from the first three scenarios against the results obtained using the methods proposed in this paper.

Table 18: Baseline Scenario Results

Scenario #	Scenario Description	Parameter Estimated	PE	"Worst Case"	"Best Case"
1	Estimate for an Agile Development Project	Delivered Size (AFP)	151	136	166
2	Estimate Effort for a Large Program	Cost ($M)	$1.5	$1.3	$1.6
3	Apply Uncertainty to a Known Productivity	Cost ($M)	$1.6	$1.4	$1.7

Table 19: "Risky Business" Scenario Results

Scenario #	Scenario Description	Parameter Estimated	PE	"Worst Case"	"Best Case"
1	Estimate for an Agile Development Project	Delivered Size (AFP)	88	34	229
2	Estimate Effort for a Large Program	Cost ($M)	$1.5	$3.7	$0.6
3	Apply Uncertainty to a Known Productivity	Cost ($M)	$1.6	$2.7	$0.5

In the case of Scenario 1, our results show a point estimate (PE) that is 42% lower and a worst case that is **75% lower**. This suggests that less code may be delivered, which is directly attributable to discrepancies in productivity uncertainty.

In Scenario 2, our result has a similar point estimate, but a worse case outcome that is **180% higher**, when productivity uncertainty is properly considered.

In Scenario 3, although we treated productivity (and, therefore, cost) as a "known value," we found it to be located at only the 32nd percentile of the fitted productivity distribution. In addition, the "worst case" cost is **92% higher** when productivity uncertainty is properly considered.

Scenario 4: Using Our Results within Monte Carlo Simulation

Under this scenario, we assume that the estimator has the same defensible productivity factor, as described in Scenario 3. We further assume that the estimator is conducting risk and uncertainty analysis using a Monte Carlo simulation. Under this approach, the analyst needs to define a probability distribution for all cost model inputs that contain uncertainty.

The application of our results to this situation is straightforward. When defining the uncertainty distribution for productivity, the analyst would select a lognormal form. The distribution parameters could be directly entered as the mean and standard deviation from Table 9 , Table 12, or Table 16, depending on whether the project size is small, medium, or large. Alternatively, if the estimator wanted to use the known productivity factor as the mean, they could calculate a CV by taking our mean divided by standard deviation. The distribution could then be defined using the known mean and calculated CV.

Scenario 5: Productivity as an Output

The final scenario that we offer is one in which productivity is not specified but is instead calculated as an output. As discussed in *Productivity as an Output*, if the estimator is using a parametric equation such as COCOMO II, then there is no direct specification of productivity.

For this scenario, we assume that the estimator has been given a KESLOC size of 200 and has also entered the E and EM parameters of COCOMO II to calculate effort of 225,175 man-hours[39].

Although productivity is implied in this process, there is no direct specification of it as an input. The analyst would calculate productivity by dividing 200,000 ESLOC by 225,175 man-hours derive a productivity factor of 0.8882 SLOC/MH.

[39] These scenario parameters, and the use of COCOMO II to calculate effort, are presented in the ICEAA Cost Estimating Body of Knowledge for Software, Lesson 4

We need to account for the fact that this productivity calculation is denominated SLOC/MH and not FP/MH. One option would be to backfire the productivity number to FP/MH using a published SLOC to FP conversion table[40]. This conversion would yield a productivity factor denominated in FP/MH, and any of the techniques in the previous scenarios could be applied. A second option would be to use our mean and standard deviation parameters to calculate a CV. Because CV is unitless, it could be used to specify a lognormal distribution using the calculated productivity as the mean. In either situation, our results would provide a valuable way to assess the implied productivity from the COCOMO II equation, and to add risk and uncertainty to the estimated effort.

Conclusions and Discussion

It's well-documented that every software development effort estimate should start with an estimate of size[41]. Beyond that, some methods also require an estimate of productivity, while others treat productivity as an output.

Assuming that size is known (or at least, has its uncertainty properly qualified), an analyst makes a significant assumption when they *also* assume a productivity level. After all, since Productivity = Size/Effort *by definition*, it follows that Effort = Size/Productivity, *also by definition*. If size and productivity are assumed, there is nothing left to estimate, and effort is also effectively assumed, rather than estimated. An estimator who assumes size and productivity values has, in effect, assumed away their estimate! In this regard, treating productivity as an input is problematic on its face.

Nevertheless, productivity-based estimating is popular and sometimes necessary. This is especially true when either a quick-turn or Rough Order of Magnitude (ROM) estimate is required, an estimate is analogy-based, or an estimate is for an agile project. When productivity is simply assumed in this manner, it is essential to quantify

[40] For example, The QSM Function Point Languages Table, https://www.qsm.com/resources/function-point-languages-table
[41] *Cost Estimating Body of Knowledge for Software* (CEBoK-S), Lesson 4.

uncertainty around it, and to consider the implications of that uncertainty on the overall cost or effort estimate. Even when productivity is calculated by a more sophisticated EER, it is important to cross-check the result against published productivity benchmarks or ranges.

In light of these critical considerations, we previously stated that our analysis would address three questions:

- How much variance is present in productivity data?
- What probability distribution should be used to model productivity?
- What distribution parameters best fit our data?

We have answered all three questions in this paper. Our analysis has shown that:

- There is tremendous variance in productivity data, with CVs exceeding 100%.
- Productivity is best modeled using a lognormal distribution.
- The best-fitting lognormal parameters vary, but only slightly, based on the “size bucket” of the project within the ISBSG dataset.

Because the lognormal distribution is supported in MS Excel, generating S-curves and cross-checks of point estimates using our fitted distributional parameters is fairly straightforward. Additionally, the real-world scenarios presented in *Practical Applications* demonstrate that use of these distributional parameters will provide a more accurate point estimate and/or a more realistic uncertainty range around an estimate of productivity, total effort, or total cost. We do not assert that Productivity follows a lognormal distribution. Rather, we assert that the lognormal distribution provides the best approximation of the distribution of productivity relative to all candidate distributions that can reasonably be considered.

The S-curves associated with our three fitted lognormal distributions are shown in the figure below. In this graph, the thickness of the curve indicates the size of the project, with the thickest curve corresponding to the largest projects in the ISBSG dataset.

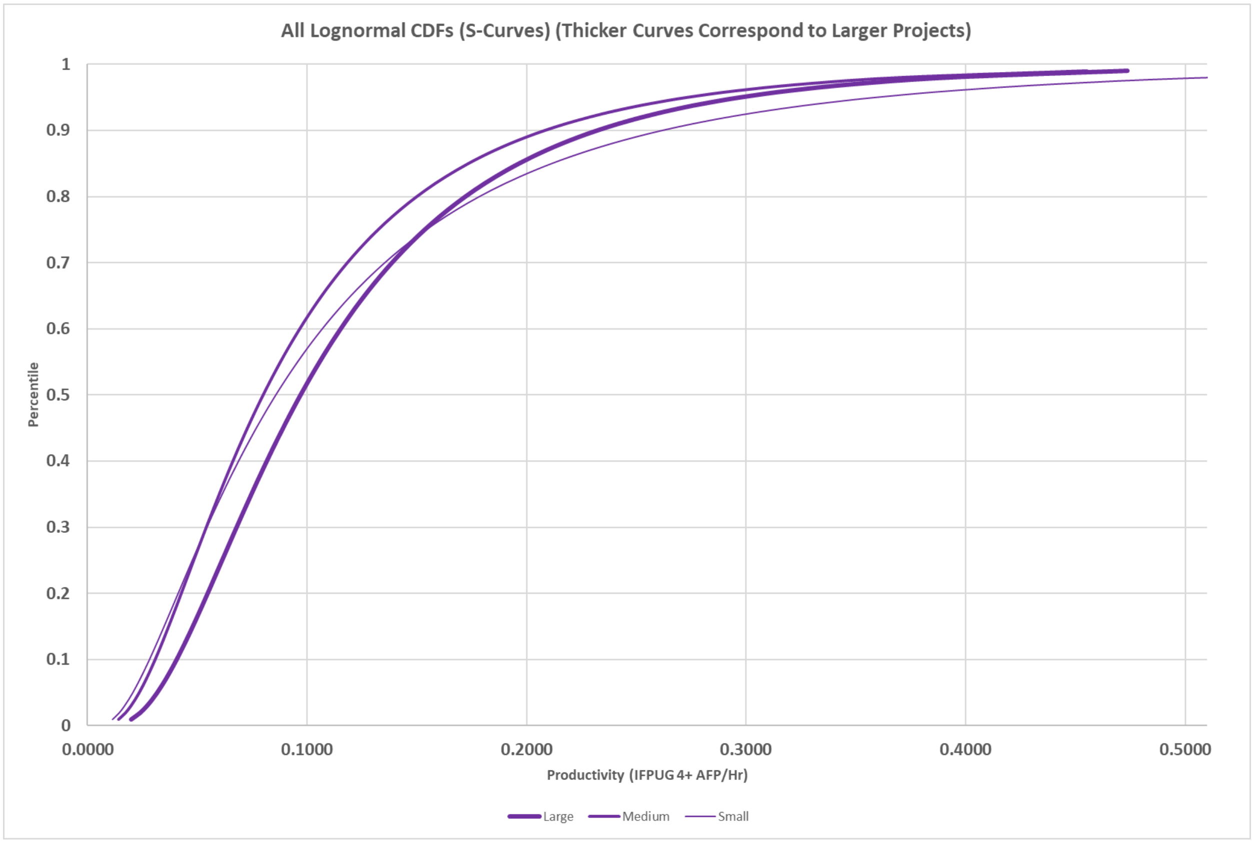


Figure 13: All Fitted Lognormal CDFs (S-Curves)

Could one have reasonably foreseen the result that productivity is best modeled as a lognormal distribution? Perhaps. Recall that, in a lognormal distribution, the distribution of the natural logarithms ("logs") of the underlying data is normal. We also know the log of a quotient equals the difference of the logs, and that the log of the difference of two normal random variables is normal. Symbolically:

$$Productivity = \frac{Size}{Effort}$$

$$LN\ (Productivity) = LN\left(\frac{Size}{Effort}\right) = LN(Size) - LN(Effort)$$

Therefore, if log-Size and log-Effort are normal, then log-Productivity must also be normal, meaning that Productivity must be lognormal. More generally, the quotient of two lognormal random variables is *itself* lognormal.

The following figures plots the frequencies of log-Size, log-Effort, and log-Productivity, respectively, based on the frequency of points by bin, relative to their percentage deviation from their range midpoints[42].

[42] This convention is used so that all three data series can be compared on the same graph.

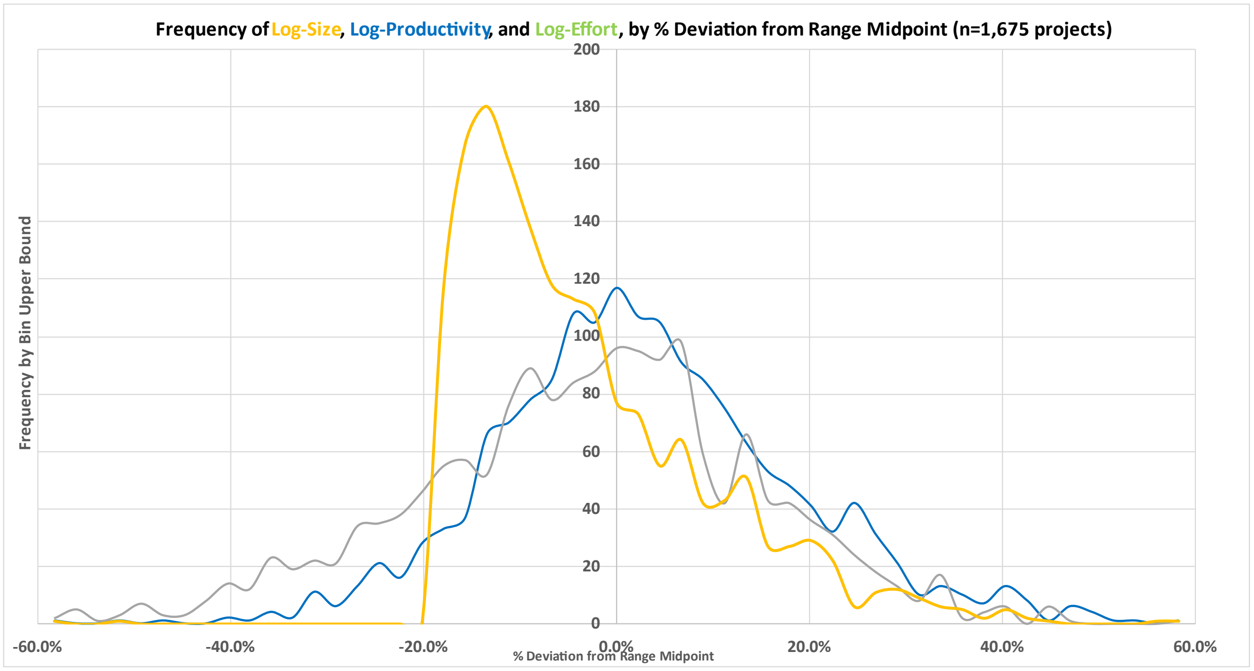


Figure 14: Frequency of Log-Size, Log-Productivity, and Log-Effort by % Deviation from Range Midpoint

This graph uses binned data to compare the distributions of log-space Size, Productivity, and Effort to each other on a common scale. For each variable, there are 53 bins, with the lower bound of the first bin corresponding to the minimum of the dataset, and the upper bound of the last bin corresponding to the maximum. This implies that the range midpoint for each variable occurs at the midpoint of the 27th bin. The graph uses the convention that “yellow (shown as orange for greater contrast) and blue make green.” Since size and productivity jointly determine effort, the three are depicted at yellow/orange, blue, and light green, respectively.

For example, on the blue (productivity) graph, the y-intercept is (0,117). This implies that, for the 27th (middle) bin, which exactly lies at the midpoint of the range of log-Productivity values, the number of projects in the bin (i.e., extremely close to the midpoint) is 117 out of 1,675.

From visual inspection, the distribution of Size is not lognormal: it is highly right-skewed, even in log space[43]. However, the distribution of Effort looks approximately normal in log space, perhaps with some slight *left*-skew. These skewness values appear to approximately offset[44], as the distribution of log-Productivity appears approximately normal (with a skewness value that implies approximate symmetry), implying that the distribution of Productivity is approximately lognormal, supporting our finding.

Next Steps

The research detailed in our paper constitutes an important advance and practical, easy-to-understand findings that can and should be applied by software cost analysts TODAY. Not surprisingly, our work on this paper has highlighted the need for future complementary research introduced below.

Our research focused on modeling the uncertainty around productivity, rather than the issue of economies of scale (EoS) vs. constant returns to scale (CRS) vs. diseconomies of scale (DoS). However, complete separation of the issues is impossible, because the EoS and DoS constructs imply that productivity changes as size changes.

The prevailing wisdom in the cost community on this topic is that DoS prevails[45], with associated slopes of the log-space regression of Effort on Size ranging from 1.0997[46] to 1.20 or more. The research giving rise to those values has primarily focused on US Department of Defense (DoD)-heavy datasets, whose projects tend to be much larger than the ones in ISBSG. Our results suggest that the usual DoS construct may be

[43] This raises an interesting possibility: is the distribution of Size log-lognormal? In other words, would twice taking the natural log of Size yield a normal distribution, so that LN(Size) is *itself* lognormal?

[44] Indeed, we calculated skewness and kurtosis (tail-heaviness) values for log-Size, log-Effort, and log-Productivity. Based on these attributes, each log-space distribution tested as normal (implying that each distribution is lognormal), except for log-Size, which has a skewness value too high (too right-skewed) to support normality. In other words, Productivity could be thought of as the ratio of a *log-lognormal* variable (Size) to a lognormal variable (Effort).

[45] See CEBoK, CEBoK-S, and others.

[46] COCOMO II default

inappropriate for smaller projects. In fact, EoS may prevail for smaller projects, with CRS and ultimately DoS prevailing as Size increases. In other words, the scale exponent *b* may *itself* vary with Size.

In the table below, we calculated *a* and *b* values for various subsets of the ISBSG dataset, fitted based on the loglinear regression Effort = a*Size^b.

Table 20: Fitted Values of a (Intercept) and b (Log-Space Slope) by Size Bucket

Size Category	AFP Range	Number of Projects (n)	b	Ln(a)	a
All data	7 to 2,048	875	0.845	3.093	22.03
Small (lowest third)	7 to 70	567	0.181	5.744	312.29
Medium (middle third)	71 to 115	549	1.131	1.882	6.57
Large (highest third)	116 to 2,048	559	0.854	3.029	20.67
Lower Half	7 to 87	839	0.680	3.754	42.71
Upper Half	88 to 2,048	836	0.776	3.473	32.22
Smallest projects	7 to 50.76	29	-0.284	7.481	1,774.66
Largest projects	522 to 2,048	30	1.081	1.570	4.81

The table suggests that *b* increases as size increases. The very smallest projects exhibit a negative *b* value[47]; intermediate projects exhibit a *b* value that is positive, but less than one (indicating EoS); the very largest projects exhibit a b value that is greater than one (indicating DoS). This is consistent with standard economics textbooks, in which EoS occurs at low quantities, slowly changing to DoS at large quantities[48]. This idea is so fundamental that it's part of almost any introductory economics curriculum. Why should software development be different? This is suggestive that the idea of DoS "everywhere and anywhere" may be misplaced. Future research could focus on exploring whether, and how, *b* varies with size.

[47] This suggests that effort declines, in absolute terms, as size increases, for very small projects. The idea is not as crazy as it may initially sound. A very small software project might have only one developer who may have weaknesses and related struggles with parts of the code. A slightly larger project may add a second developer, where each developer can focus on their respective strengths, so that total effort is lower for the second project.

[48] See, for example, https://www.repetico.com/card-72057429, for a homework-oriented "flashcard" demonstrating this idea.

This future research would be especially important for ICEAA, as it grows its international stakeholder base, since many non-US projects are smaller than traditional DoD-oriented projects.

Acronyms and Abbreviations

(A)FP	(Adjusted) Function Points
CADE	Cost Assessment Data Enterprise
CDF	Cumulative Distribution Function
CEBoK(-S)	Cost Estimating Body of Knowledge (for Software)
COCOMO	Constructive Cost Model
COSMIC	Common Software Measurement International Consortium
CRS	Constant Returns to Scale
CSS	Chi-Square Statistic
CV	Coefficient of Variation
DoD	Department of Defense
DoS	Diseconomies of Scale
EER	Effort Estimating Relationship
EM	Effort Multiplier
(E)SLOC	(Equivalent) Source Line(s) of Code
EoS	Economies of Scale
EVM	Earned Value Management
FPA	Function Point Analysis
ICEAA	International Cost Estimating and Analysis Association
IFPUG	International Function Point Users Group
ISBSG	International Software Benchmarking Standards Group
LN	Natural Logarithm
MH	Man-Hour(s)
MLE	Maximum Likelihood Estimation
MoM	Method of Moments
MS	Microsoft
NCCA	Naval Center for Cost Analysis
P/E	Price to Earnings
PDF	Probability Density Function
PDR	Project Delivery Rate
PERT	Program Evaluation Review Technique
PM	Person-Month(s)
ROI	Return on Investment
ROM	Rough Order of Magnitude
SME	Subject Matter Expert
SSE	Sum of Squared Error
StDev	Standard Deviation
SWEET	Software Effort Estimating Tool