# ASCoT 3: Nonlinear Principal Components Analysis and Uncertainty Quantification in Early Concept Spacecraft Flight Software Cost Estimation

**Sam Fleischer, Patrick Bjornstad, and Jairus Hihn**
**Jet Propulsion Laboratory,**
**California Institute of Technology**
**Pasadena, CA 91109**

**James Johnson***
**National Aeronautics and Space Administration**
**Washington, DC 20546**
**james.k.johnson@nasa.gov**

***Abstract*—For mission planners and evaluators alike, value in cost models comes from a mean or median prediction, an understanding of the uncertainty on that prediction, and an understanding of model performance. Here we apply advanced statistical and machine learning methods to spacecraft flight software cost, effort, and SLOC estimation, and present the results in the latest version of the Analogy Software Cost Tool (ASCoT). We present in- and out-of-sample performance metrics for our models, each of which incorporate some amount of epistemic uncertainty. ASCoT, hosted on the One NASA Cost Engineering (ONCE) database via the Online NASA Space Estimation Tool (ONSET), was first showcased in 2016 as a number of analogy-based models and methods (*k*NN and Clustering) to support early project formulation. This ASCoT update improves upon the previous analogic methods by incorporating uncertainty in the data transformations. In particular, we use a Nonlinear Principal Components Analysis (NLPCA) to deal with ordinal data.**

### TABLE OF CONTENTS



## 1. INTRODUCTION

Proposal concepts and designs need to be evaluated quickly and early in the lifecycle in order to keep pace with the increasing number of proposal calls released by NASA. Mission planners and evaluators use parametric and nonparametric cost models to assess concepts' most pressing cost and risk drivers. In particular, sensitivity studies are routinely performed on new and established models to quantify how the mean or median cost predictions respond to architectural changes. As models improve in accuracy and precision, fewer fruitless architectural decisions are made early in the project lifecycle, saving time, decreasing cost, and increasing the probability of success [1].

Here we present novel models and methods for analogy-based cost estimation for spacecraft flight software. We implement a Nonlinear Principal Components Analysis (NLPCA) algorithm with epistemic uncertainty to deal with non-numeric data and produce probabilistic estimates for mean cost (in the form of labor effort and number of lines of code [2, 3]) alongside probabilistic statements about analogies. We also improve the previous ASCoT Cost Estimating Relationships (CERs) by modeling the entire posterior distribution of parameters of a Bayesian regression, and by using out-of-sample model performance metrics for model selection, to produce a rule of thumb with aleatoric and epistemic uncertainty for total flight software cost as a function of spacecraft bus cost.

ASCoT was first showcased at the 2016 IEEE Aerospace conference [4-6], where the importance of using a range of validated models in the early lifecycle in order to minimize risk of cost overruns was elucidated [1]. Here, we continue to advocate for the simultaneous usage and advancement of multiple methodologies to predict cost, as well as the explicit inclusion of uncertainty in cost and schedule estimates.

Section 2 contains detailed descriptions of the advanced methods we use, including the Bayesian regression with Hamiltonian Markov Chain Monte Carlo, NLPCA, *k*NN, and Clustering. In Section 3 we explain the model performance metrics used for model selection, and how our models perform with respect to those metrics. We close out with discussion in Section 4.

## 2. MODELS AND METHODS

### *A. Bayesian Regression*

Here we present a simple parametric model which predicts software cost using only the cost of a spacecraft bus and its orbital destination. We use the following model form:

$$\log(\mathrm{Cost}_{\mathrm{FSW}}) \sim \mathrm{SkewNormal}(\mu, \sigma, \alpha),$$
$$\mu = \beta_0 + \beta_1 \log(\mathrm{Cost}_{\mathrm{SC}}) + (\beta_2 + \beta_3 \log(\mathrm{Cost}_{\mathrm{SC}}))\mathbf{1}_{\mathrm{Earth}}, \quad (1)$$

where $\mathrm{Cost}_{\mathrm{FSW}}$ is the Cost of the flight software, $\mathrm{Cost}_{\mathrm{SC}}$ is the cost of the spacecraft bus, $\mathbf{1}_{\mathrm{Earth}}$ is an indicator variable

978-1-6654-9032-0/23/$31.00 ©2023 IEEE

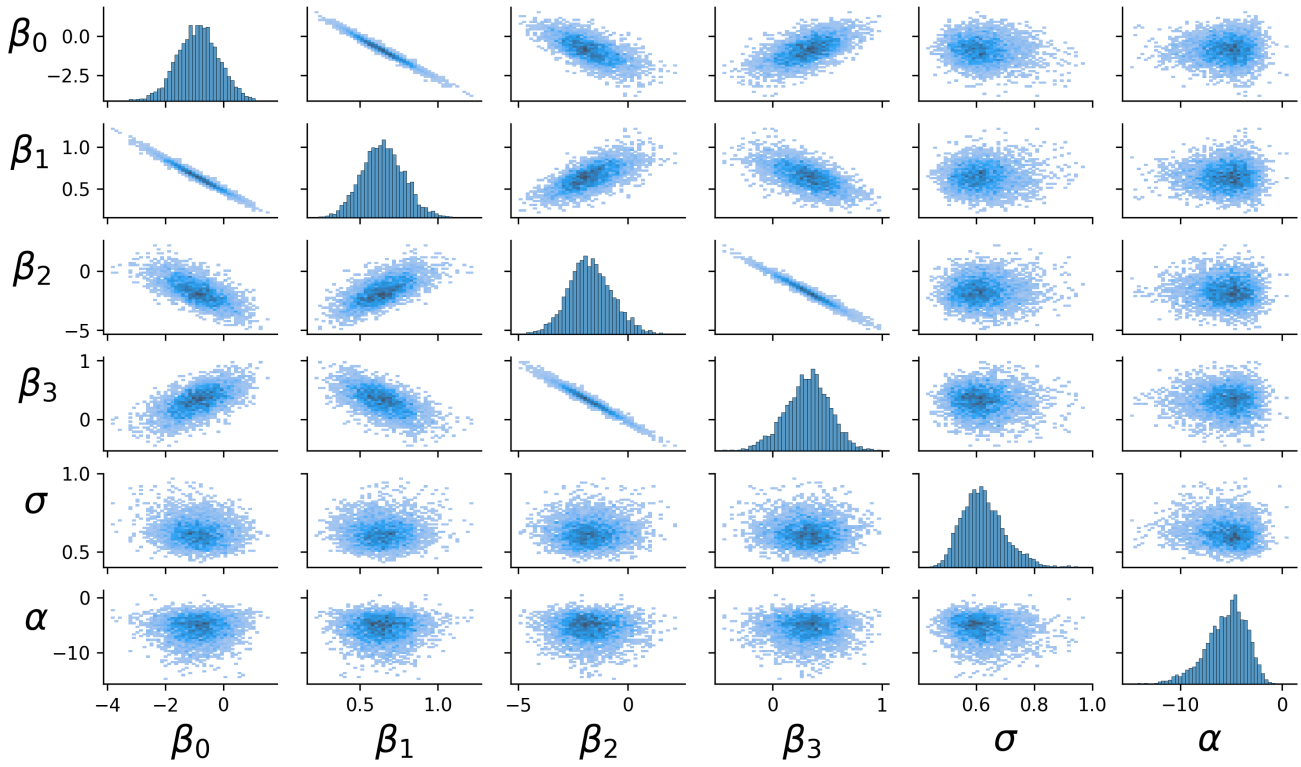


**Figure 1. Pairplot for the posterior distribution of the parameters in Equation (1).**

equal to 1 if the spacecraft's destination is Earth and 0 otherwise, and $\mu, \sigma^2, \alpha$ are the mean, variance, and skew of the Skew Normal distribution. Using uninformative priors and the ASCoT CER dataset (N=43), we estimate the six-dimensional posterior distribution of $\sigma^2$, $\alpha$, $\beta_0$, $\beta_1$, $\beta_2$, and $\beta_3$ using the probabilistic programming language Stan [7]. Figure 1 shows the posterior distribution of the parameters in Equation (1).

Typical linear regression assumes a normal distribution around a mean line. Here we choose a skew normal distribution to better align with our data. More specifically, there are a few missions with very cheap flight software costs relative to their spacecraft bus costs. Analysis of a linear regression with a normally distributed error term might suggest these points are outliers and analysts may choose to remove those points from the dataset. Analysis of the skew normal model suggests the more predictive error distribution around the mean cost has more density at lower costs because of the negative skew parameter $\alpha$. Of course, adding in an additional parameter increases the risk of overfitting, but we use out-of-sample model performance metrics to confirm that the skew parameter adds predictive power and is not more overfit than a typical regression (see Section 3).

Note also the marginal distribution of the parameter $\beta_3$ overlaps with zero. Analysts using typical linear regression may come to the conclusion that $\beta_3$ is not significant and decide to remove this term because the $p$-value may be greater than 0.05. But this model performs better *predictively* than the model without the $\beta_3$ term, based on a leave-one-out cross validation (see Section 3).

### *B. Nonlinear Principal Components Analysis*

NASA cost analysts are routinely gifted datasets with many fields but few records. The most pressing question is, of course, how to find the variable or group of variables that best predict cost or some other variable of interest. A good starting point is to fit a sequence of simple and multiple regressions and comparing models along the way.

A next step might be to fit a PCA regression, in which a (linear) Principal Components Analysis is done on the continuous predictor variables in the dataset. Because principal components from a PCA analysis are orthogonal, there is no risk of misinterpreting results from models with correlated variables. But if there are no (or few) continuous variables in the dataset, or you suspect the features of the data may be nonlinear, then PCA will be insufficient and misleading.

In these all-too-common scenarios, some sort of nonlinear dimension reduction scheme must be deployed. For a dataset of predictors $X$ with domain $\Omega$ of dimension $k$ to be optimally nonlinearly reduced to dimension $\ell < k$, we must determine functions $f: \Omega \to \mathbb{R}^\ell$ and $g: \mathbb{R}^\ell \to \Omega$ such that $\|X - g \circ f(X)\|$ is minimized. In other words, we want an autoencoder $g \circ f$ from $\Omega$ to $\Omega$, with a bottleneck of dimension $\ell$. Intuitively, if data of dimension $k$ can be sufficiently recovered after passing through this low-dimensional bottleneck, then we have an acceptable

dimension reduction function $f$ (also called the "mapping function") and recovery function $g$ (also called the "de-mapping function").

One natural way to learn the mapping and de-mapping functions are with feed-forward neural networks (FFNNs). FFNNs are built up with perceptrons, which are simply nonlinear, typically nondecreasing functions applied to linear combinations of variables. Mathematically, a perceptron is a function with domain $\mathbb{R}^n$ and codomain $\mathbb{R}$ of the form

$$Y = f\left(b + \sum_{i=1}^{n} w_i X_i\right) = f\left(b + \vec{w} \cdot \vec{X}\right),$$

where $b$ is a bias parameter and $\vec{w}$ is a vector of weights. A *layer* in a FFNN is a vector of perceptrons with the same domain. Mathematically, a layer is a function with domain $\mathbb{R}^n$ and codomain $\mathbb{R}^m$ of the form

$$\vec{Y} = F\left(\vec{b} + \overrightarrow{\overrightarrow{W}} \cdot \vec{X}\right),$$

where $\vec{b}$ is a vector of bias parameters and $\overrightarrow{\overrightarrow{W}}$ is a matrix of weights. A FFNN is a sequence of layers, each successive layer's domain equaling the previous layer's codomain.

We can specify the form of a FFNN using a list of dimensions of the input and each successive layer. For example, if the dimension of the input is 7 and the dimension of the output is 3, then the FFNN has one layer, and we say the FFNN has dimension [7,3]. FFNNs with multiple layers are specified using longer lists; for example, a FFNN with dimension [7,8,2,8,7] has four layers – the input and output are of dimension seven, and the intermediate steps have dimension eight, two, and eight, respectively.

Fitting a FFNN to data means finding the parameters of the FFNN (the weight matrix $\overrightarrow{\overrightarrow{W}}$ and the bias vector $\vec{b}$ for each layer) to minimize predictive error. This is typically done using an efficient algorithm called back-propagation, which optimizes subsets of the FFNN in sequence instead of the entire FFNN at the same time using classical optimization schemes like gradient descent.

For nonlinear dimension reduction, ASCoT uses FFNNs to generate the mapping and de-mapping functions. In particular, we fit FFNNs with dimension $[D, x, d, x, D]$, where $x$ is some positive integer greater than $d$, $D$ is the dimension of the data (in our case 6 or 7), and $d$ is the number of nonlinear principal components we are interested in finding (in our case 2). After fitting the FFNN to the data, the mapping function is the first half of the FFNN, of dimension $[D, x, d]$, and the de-mapping function is the second half of the FFNN, of dimension $[d, x, D]$.

Future ASCoT work could involve more state-of-the-art dimension reduction algorithms like KernelFlows, in which the mapping and de-mapping functions are defined as Gaussian processes rather than FFNNs.

There are fewer data records than parameters in the defined FFNNs, which causes an identifiability problem when fitting the mapping and de-mapping functions. In other words, there is not an optimal single set of weights and biases, but rather an optimal manifold of weights and biases. In order to account for the uncertainty of weights and biases, we fit 1000 independent FFNNs with random initializations, and therefore extract 1000 paired mapping and de-mapping functions. This means that instead of a single low-dimensional dataset, we have a distribution of low-dimensional datasets that we are able to analyze. The distributions are used in our *k*-Nearest Neighbors and Clustering algorithms.

### *C.* k-*Nearest Neighbors*

The ASCoT *k*-Nearest Neighbors (*k*NN) algorithm is a nonparametric algorithm to predict the Effort required to produce flight software, in work-months (WM), or the number of thousands of equivalent logical source lines of code (kEqSLOC) in completed flight software, based on input proximity to existing data records. For the ASCoT *k*NN Effort model, users input the following variables:

1. **Inheritance Level**: the proportion of the flight software codebase inherited from past projects,
2. **Mission Size**: the cost target (including development and operations) of the entire mission,
3. **Mission Type**: the top-level architecture of a mission (orbiter, observatory, lander, rover),
4. **Redundancy**: the architecture of backup computers on board the spacecraft,
5. **Destination**: the orbital or planetary destination of the mission,
6. **Number of Instruments**: the number of scientific instruments on board the spacecraft, and
7. **Number of Deployables**: the number of unique non-scientific, movable components on board the spacecraft (booms, arms, etc.).

For the ASCoT kEqSLOC model, users input the same variables except the inheritance level.

Let $\vec{x}_u$ denote the vector of user input variables, and let $f(\vec{x}_u)$ denote the dimension-reduced, numerically transformed user input after passing through the same numerical transformations as the data (N=39 for the Effort model and N=46 for the kEqSLOC model). We calculate the distance $d_i$ to $f(\vec{x}_u)$ from each transformed record in the dataset $f(\vec{X}_i)$, defined

$$d_i = \left\| f(\vec{x}_u) - f(\vec{X}_i) \right\|_2,$$

where $\|\cdot\|_2$ is the Euclidean norm. The $k$ data points $\vec{X}_i$ closest to the input in the transformed space (smallest $d_i$) are called the "$k$ nearest neighbors." After reordering the points $\vec{X}_i$ such that $d_1 \leq d_2 \leq \cdots$ (i.e. in order from closest to farthest), we estimate Effort or kEqSLOC as the weighted average

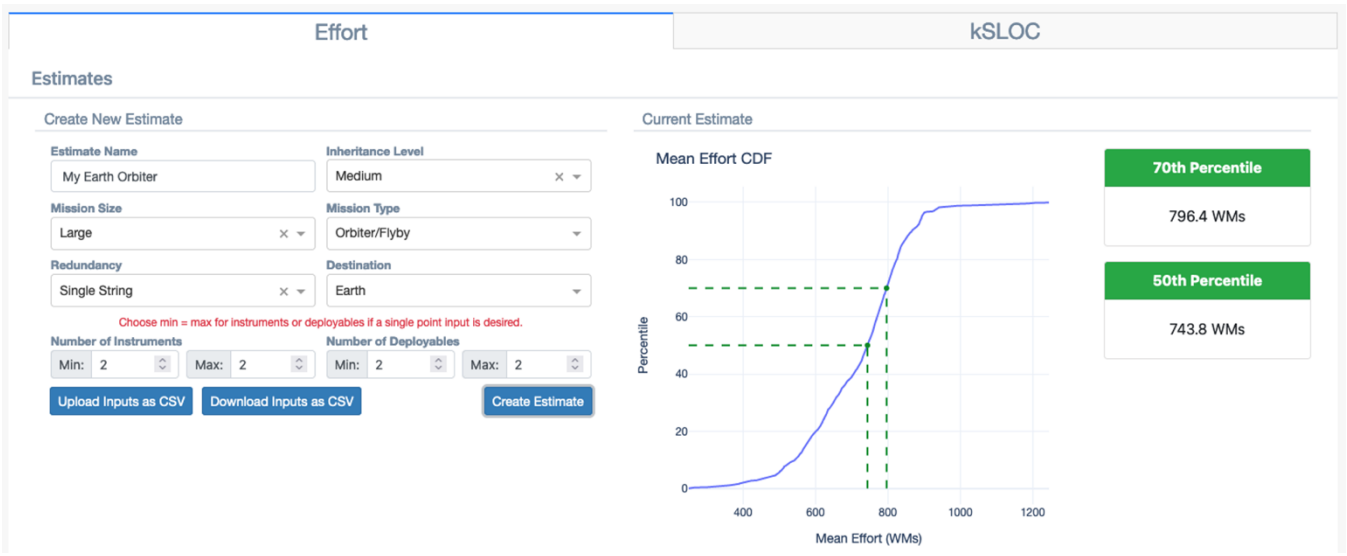


**Figure 2. ASCoT Effort *k*NN tool inputs (left) and Mean Effort distribution (right). The 50th and 70th percentiles of the distribution are highlighted. Inputs are "Medium" inheritance (20-50%), "Large" mission size (600M-1.1B in $FY16), "Orbiter/flyby" mission type, "single string" redundancy, "Earth" destination, two instruments, and two deployables**

$$y_u = \frac{\sum_{i=1}^{k} \frac{Y_i}{d_i}}{\sum_{i=1}^{k} \frac{1}{d_i}}, \qquad (2)$$

where $Y_i$ denotes the Effort or kEqSLOC associated with $X_i$.

Previous versions of ASCoT reported the weighted average and which missions were nearest. In the newest version of ASCoT, we have a distribution of data transformations, which means we compute a distributon of weighted averages as well as a distribution of neighbors. The current ASCoT version reports the full distribution of mean Effort and kEqSLOC, and, for each mission in the dataset, the probability it is one of the three nearest neighbors. Figure 2 shows an example of user input and the resulting distribution of Effort, and Figure 3 shows the probabilities each mission is one of the three nearest neighbors. Users also have the option to leave certain inputs blank if they are uncertain and give ranges for the number of instruments and deployables.

### *D. Clustering*

Like *k*NN, the ASCoT Clustering algorithm is another nonparametric algorithm to predict Effort or kEqSLOC. Like the *k*NN algorithm, user input is transformed according to the nonlinear dimension reduction mapping functions described above. The input is assigned to a particular cluster of missions based on proximity to the cluster. Then Effort or kEqSLOC is estimated as a weighted average of the missions in the cluster. Specifically, if the transformed input $f(\vec{x}_u)$ is placed in cluster $j$ and $C_j$ is the collection of missions in cluster $j$, then Effort or kEqSLOC is:

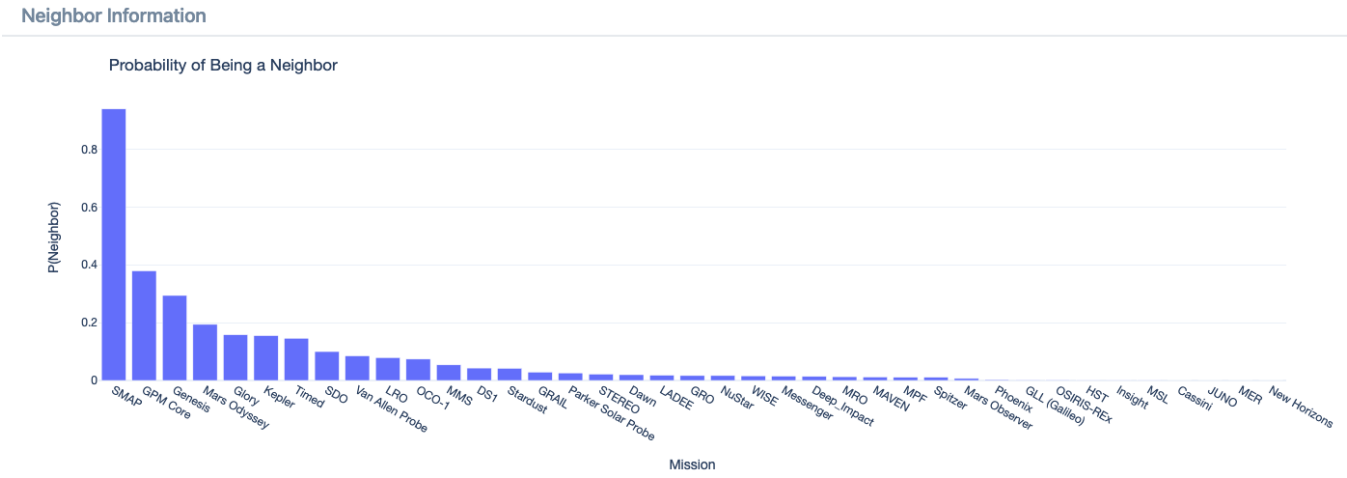


**Figure 3. ASCoT Effort *k*NN tool probabilities of being a neighbor, given inputs defined in Figure 2. SMAP is very likely the most useful analog, and other potentially useful analogs are GPM Core and Genesis.**

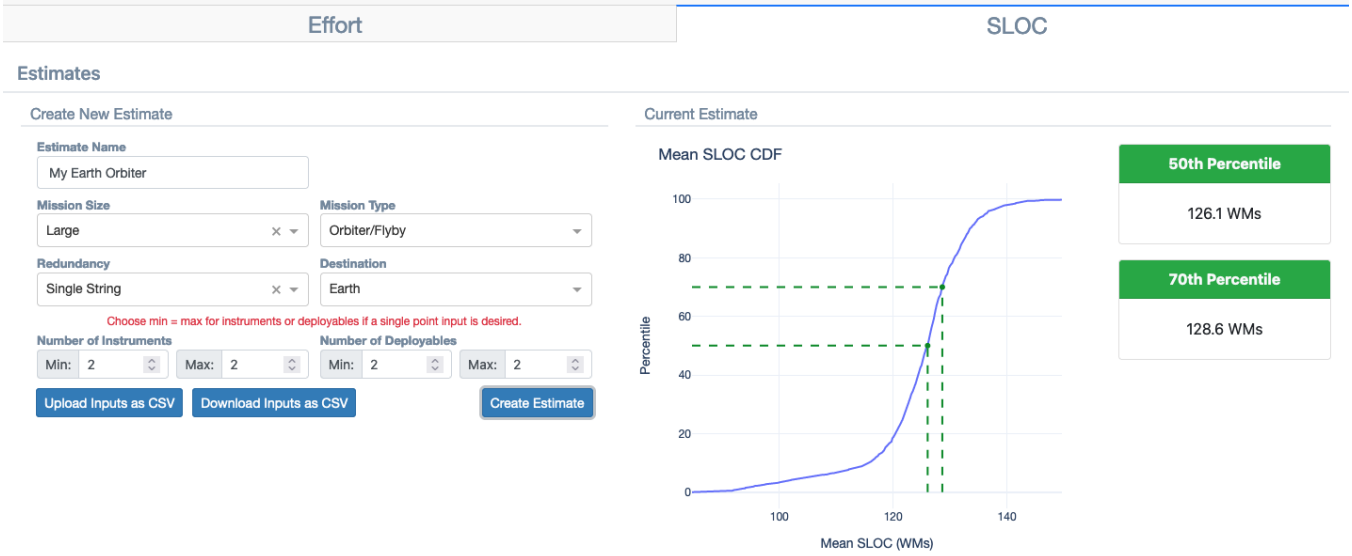


**Figure 4. ASCoT kEqSLOC Clustering tool inputs (left) and Mean kEqSLOC distribution (right). The 50th and 70th percentiles of the distribution are highlighted. Inputs are "Large" mission size (600M-1.1B in $FY16), "Orbiter/flyby" mission type, "single string" redundancy, "Earth" destination, two instruments, and two deployables.**

$$y_u = \frac{\sum_{c \in C_j} \frac{Y_c}{d_c}}{\sum_{c \in C_j} \frac{1}{d_c}}, \quad (3)$$

where $Y_c$ denotes the Effort or kEqSLOC associated with mission $c$ and $d_c = \|f(c) - f(\vec{x}_u)\|_2$ is the Euclidean distance between mission $c$ and the user input in transformed space. Also like the *k*NN algorithm, since there are a distribution of transformations, we calculate a distribution of weighted averages as well as the distribution of cluster assignments. Figure 4 shows an example of user input and the resulting distribution of kEqSLOC, and Figure 5 shows the probability distribution of cluster assignment, as well as the kEqSLOC distribution colored by cluster assignment.

## 3. Model Performance

After building and fitting one or more models, it is critical to then assess model performance. The most useful metrics for assessing performance are those which account for both fit to the data *and* out-of-sample prediction. Especially in the context of small sample size, analysts who fit statistical models and report $R^2$ only are potentially endorsing models which are overfit to the data and have very bad predictive performance. Intuitively, for predictive models, we are less interested in how well a model fits the data you have, and more interested in how well a model predicts data you do not have. We suggest instead using cross validation techniques to estimate out-of-sample performance. In particular, the expected log predictive density, or *elpd*, is the theoretical expected log pointwise predictive density for a

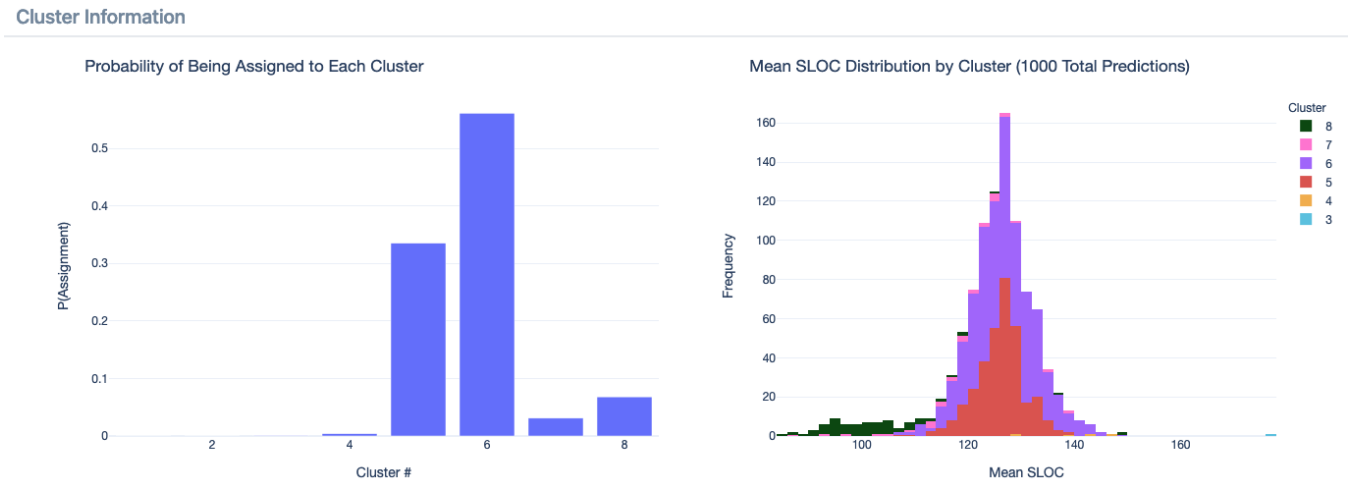


**Figure 5. ASCoT kEqSLOC Clustering tool probabilities of being assigned to a cluster given inputs defined in Figure (left) and the stacked histogram of mean kEqSLOC colored by cluster (right). Clusters 5 and 6 are the most likely assignments.**

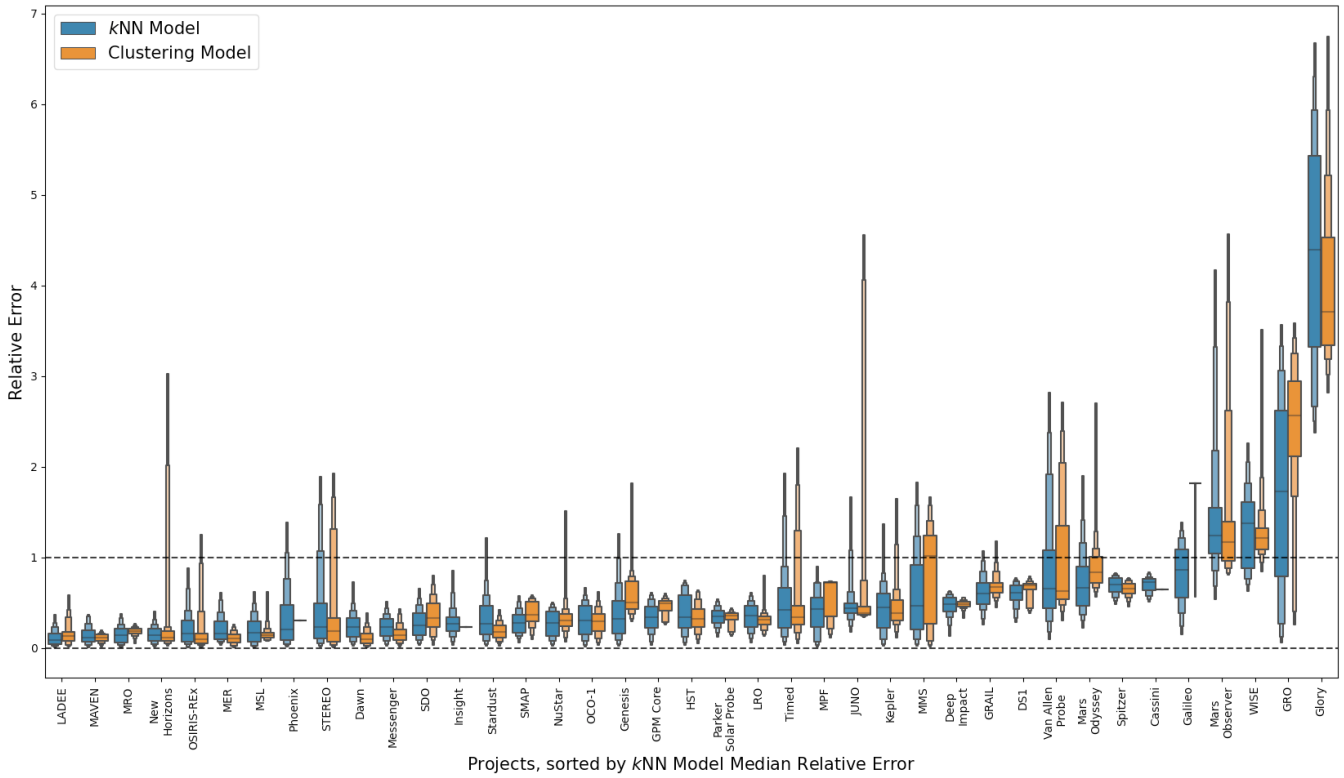


**Figure 6. Relative Error of the *k*NN and Clustering Effort models for each of the 39 projects used to fit the models, ordered by median *k*NN Effort model performance.**

new dataset, which is estimated using a leave-one-out cross validation (LOOCV) scheme, and denoted $\text{elpd}_{\text{loo}}$:

$$\text{elpd}_{\text{loo}} \coloneqq \sum_{i=1}^{n} \log p(y_i \mid y_{-i}), \qquad (4)$$

where $p(y_i \mid y_{-i})$ is the predictive density of the $i$th data record given the data without the $i$th data record (leave-one-out), but rather than refitting the model without each data record, we can estimate $\text{elpd}_{\text{loo}}$ using importance sampling [8]. $\text{elpd}_{\text{loo}}$ for our CER model is approximately $-39.1$. The $\text{elpd}_{\text{loo}}$ for the model without an indicator variable for Destination is approximately $-39.3$, and the standard error of the difference of $\text{elpd}_{\text{loo}}$ between the two models is 1.9.

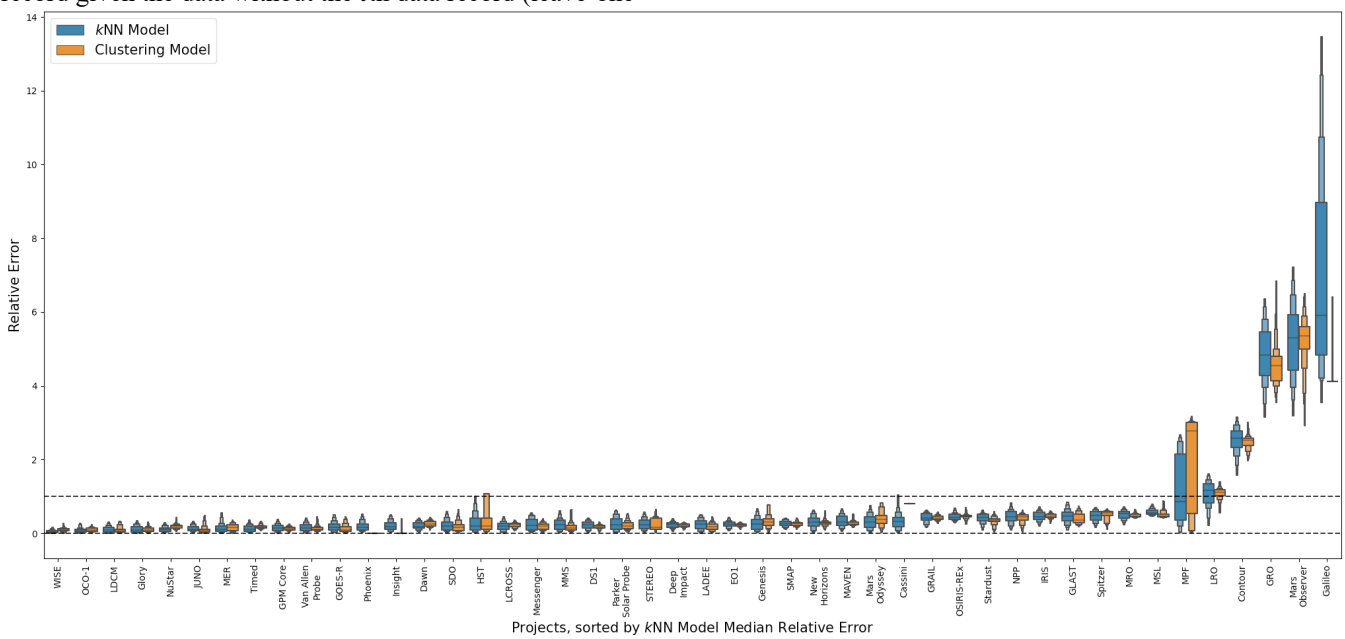


**Figure 7. Relative Error of the *k*NN and Clustering kEqSLOC models for each of the 46 projects used to fit the models, ordered by median *k*NN kEqSLOC model performance.**

Thus there is only a weak signal that the model containing the indicator variable for Destination performs predictively better than the model without it, and therefore ASCoT developers should consider removing Destination from the CER in future releases.

Figures 6 and 7 show the relative predictive errors for each point the Effort and kEqSLOC *k*NN and Clustering models, respectively. 10.3% of the points in the *k*NN Effort model, 12.8% of the points in the Clustering Effort model, 10.9% of the points in the *k*NN kEqSLOC model, and 13.0% of the points in the Clustering kEqSLOC model have MdRE > 1, respectively, which is a significant improvement over the most recent previous version of the ASCoT analogic models [5]. Also, 50% of the MdRE values for the *k*NN Effort, Clustering Effort , *k*NN kEqSLOC, and Clustering kEqSLOC models are below 0.35, 0.36, 0.25, and 0.23, respectively, all improvements over the previous ASCoT version at 0.36, 0.36, 0.34, and 0.34, respectively.

## 4. Conclusions and Discussion

The Analogy Software Cost Tool (ASCoT) contains multiple tools to help cost analysts predict the cost of spacecraft flight software. We provide four non-parametric models (*k*NN and Clustering for Effort and kEqSLOC) and two parametric models (COCOMO-II and the CER). This paper has detailed significant improvements in model performance and uncertainty quantification and communication for the *k*NN, Clustering, and CER models.

The ASCoT 3 Bayesian regression model (the CER) represents an improvement over typical linear regression because (a) the model was chosen based on out-of-sample predictive performance, and not fit to the data alone, (b) predictions come from sampling the full posterior distribution of parameters and not an assumption of multivariate normality or orthogonality, and (c) we use the skew normal error distribution instead of the typical normal distribution, which accounts for what would have been considered outliers in the dataset. The online tool allows users to download the full posterior distribution of parameters and specify inputs with different types of uncertainty. The output in the tool is shown either in a log-log scale or in the native space.

All four ASCoT 3 nonparametric models (*k*NN and Clustering Effort and kEqSLOC) now incorporate uncertainty inherent in the nonlinear dimension reduction, which propogates to uncertainty in the mean Effort and kEqSLOC predictions and in the probability distribution of neighbors and clusters. Analysts are now better able to assess the risk of overrunning cost and determine which groups of missions should be considered analogs of the mission concept they are trying to analyze. The online tool allows users to download the raw data and the distribution of datasets defined by the nonlinear dimension reduction.

Understanding the uncertainty in cost predictions is critical in early project formulation. Without this information, project managers are unable to make data-informed decisions regarding the level of risk they are taking on. Analysts should also report how much uncertainty they believe comes from lack of knowledge about the cost drivers (epistemic uncertainty) vs uncertainty that comes from inherent randomness (aleatoric uncertainty).

## Acknowledgements

The research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration. We thank Sherry Stukes, Melissa Hooke, Zaki Hasnain, Matt Hart, Joe Mrozinski, Mike DiNicola, Michael Saing, Anto Kolanjian, and Ross Weidman for their input and support.



## References

[1] Hihn, J., Juster, L., Johnson, J., Menzies, T., & Michael, G. (2016, March). Improving and expanding NASA software cost estimation methods. In 2016 IEEE Aerospace Conference (pp. 1-12). IEEE.

[2] Clark, B., McCurley, J., & Zubrow, D. (2015). DoD Software Factbook, Version 1.1. Software Engineering Measurement and Analysis Group, Software Engineering Institute, Carnegie Mellon University.

[3] Boehm, B., Abts, C., & Chulani, S. (2000). Software development cost estimation approaches—A survey. Annals of software engineering, 10(1), 177-205.

[4] Hihn, J., Youmans, T., Lumnah, A., Saing, M., Huntington, E., Hooke, M., ... & Menzies, T. (2019, March). ASCoT, the NASA Analogy Software Cost Tool Suite: Expanding Our Estimation Horizons Jet Pro-pulsion Laboratory, California Institute of Technology Pasadena, CA 91109. In 2019 IEEE Aerospace Confer-ence (pp. 1-25). IEEE.

[5] Fleischer, S., Hihn, J., Mrozinski, J., Hooke, M., Hasnain, Z., Stukes, S., & Johnson, J. (2021, March). Online NASA Software Estimating Tools (ONSET): A Suite of Web-Based Cost Analysis Tools. In 2021 IEEE Aerospace Conference (50100) (pp. 1-16). IEEE.

[6] Fleischer, S., Hihn, J., Johnson, J. (2022, September). Advanced Statistical Methods in Spacecraft Flight Software Cost Estimation: Bayesian Regression and Nonlinear Principal Components Analysis to Support System

Engineering in the Early Project Lifecycle. 2022 INCOSE International Symposium (pp. 468-482).

[7] Carpenter, Bob, et al. "Stan: A probabilistic programming language." *Journal of statistical software* 76.1 (2017).

[8] Vehtari, A., Gelman, A. & Gabry, J. Practical Bayesian model evaluation using leave-one-out cross-validation and WAIC. *Stat Comput* **27**, 1413–1432 (2017). https://doi.org/10.1007/s11222-016-9696-4

## BIOGRAPHY

***Sam Fleischer*** *(PhD, Applied Mathematics, UC Davis, 2020) is a Systems Engineer at the Jet Propulsion Laboratory. His work includes planetary protection modeling for early mission concepts to icy moons, statistical modeling risk analysis in sample depot drops for Perseverance Rover operations, and improving uncertainty quantification using Bayesian and machine learning methods in early formulation resource models. Sam is the project manager for the Analogy Software Cost Tool. Update: Sam left JPL in January 2023 to work as a Senior Quantitative Analyst in baseball operations for the Los Angeles Dodgers.*

***Patrick Bjornstad*** *is a Systems Engineering intern at the Jet Propulsion Laboratory. He is the lead software engineer for the Analogy Software Cost Tool and supports tool development and deployment for all web-based early formulation tools at JPL. He is currently a graduate student at the University of Southern California.*

***Jairus Hihn*** *(PhD University of Maryland) is a principal member of the engineering staff at the Jet Propulsion Laboratory and the manager of the Systems Modeling and Analysis Group and is currently leading a laboratory wide cost improvement task. He has been developing estimation models and providing software and mission level cost estimation support to JPL's and NASA since 1988. Jairus is the original creator of ASCoT and SCAT.*

***James Johnson*** *is responsible for providing Cost Estimates and Assessments, Schedule Estimates and Assessments, Risk Analyses, and Joint Cost Schedule Risk Analysis for the OCFO Strategic Investments Division (SID) at NASA Headquarters. His work for NASA HQ includes supporting high level Agency studies, providing support and consultation to projects, and developing policy and guidance for the Agency in the areas of cost, schedule, and risk assessments.*