# Penalized-Constrained Regression

Combining Regularization and Domain Constraints for Cost Estimation

Kevin Joy Max Watstein

2026-02-05

## Table of contents

## Abstract

Small datasets with intercorrelated predictors pose serious challenges to Ordinary Least Squares (OLS) regression, often producing coefficients that are statistically unstable and economically implausible. A motivating example in cost estimation is the learning curve with rate effect, where lot midpoint correlates with lot size as production ramps up, and domain knowledge establishes that learning and rate slopes should be $\leq 100\%$.

This paper combines penalized regularization (Lasso, Ridge, Elastic Net) with constrained optimization to ensure economically valid coefficients while maintaining predictive accuracy. Through a Monte Carlo simulation study of 8,100 scenarios using quantity profiles randomly selected from Selected Acquisition Reports (SARs) with simulated average unit costs at varying coefficients of variation, we find that Penalized-Constrained Regression with GCV selection (PCReg-GCV) outperforms OLS in predictive accuracy, with the strongest advantage precisely when OLS produces economically unreasonable coefficients.

**Key Findings:**

- PCReg-GCV provides better out-of-sample predictions even when OLS produces reasonable coefficients
- OLS with learning variable only (OLS-LearnOnly) performs poorly outside the training range
- Constraints introduce beneficial bias that improves extrapolation
- Generalized Cross-Validation (GCV) enables stable hyperparameter selection with as few as 5 observations

The research team developed the `penalized-constrained` Python package specifically for the cost estimating community after finding no existing library that combined these capabilities.

### 0.0.1 Methodological Foundation

The objective function minimized by PCReg combines a loss function with regularization penalties subject to coefficient bounds:

$$\min_{\theta} \underbrace{\sum_{i=1}^{n} L(y_i, \hat{y}_i)}_{\text{Loss (e.g., SSPE)}} + \underbrace{\alpha \left[\rho\|\theta\|_1 + \frac{1-\rho}{2}\|\theta\|_2^2\right]}_{\text{Elastic Net Penalty}}$$

subject to: $\theta_{\text{lower}} \leq \theta \leq \theta_{\text{upper}}$

where $\alpha$ controls penalty strength and $\rho$ balances L1 (Lasso) vs L2 (Ridge) regularization.

**Bounds can be loose or tight:**

- **Loose bounds** (e.g., 70-100%): Allow flexibility while preventing egregious violations
- **Tight bounds** (e.g., 85-95%): Incorporate strong prior knowledge but risk over-constraining

Research foundations include James, Paulson, and Rusmevichientong (2020) demonstrating that constrained Lasso can achieve optimal prediction under monotonicity assumptions, and classical Ridge regression theory showing that some bias can reduce overall mean squared error when predictors are correlated.

## 0.1 Introduction

Developing Cost Estimating Relationships (CERs) for small datasets (5-30 data points), a recurring pattern emerges: strong fit statistics ($R^2$, CV) but nonsensical coefficients—wrong signs, implausible magnitudes, and poor p-values for critical predictors. As Department of Defense analysts, this story may feel all too familiar.

Multicollinear datasets are a frequent presence in cost analysis, causing models to misbehave. The consequences of multicollinearity in small samples are well-documented (Flynn and James 2016):

- **Unstable coefficient estimates**: Coefficients swing wildly with small data changes
- **Wrong coefficient signs**: Estimates flip positive/negative contrary to domain knowledge
- **Unreliable hypothesis testing**: High F-statistic but individually insignificant t-statistics
- **Inflated variance**: Coefficient variance increases by factor $1/(1-R^2)$ where $R^2$ is the correlation between predictors

### 0.1.1 The Learning Curve Problem

The motivating example for this research is the learning curve with rate effect:

$$Y = T_1 \cdot (\text{LotMidpoint})^b \cdot (\text{LotQuantity})^c \cdot \varepsilon \tag{1}$$

where:

- $T_1$ = theoretical first unit cost
- $b$ = learning slope (learning rate = $2^b$, typically 70-95%)
- $c$ = rate slope (rate effect = $2^c$, typically 70-95%)
- $\varepsilon$ = multiplicative error

In this specification, lot midpoint (learning variable) is inherently correlated with lot size (rate variable) as production ramps up. Domain knowledge establishes that learning and rate slopes should be $\leq 100\%$—costs should not *increase* with cumulative production experience.

### 0.1.2 Research Contribution

This paper provides a practical guide combining penalized regularization with constrained optimization for cost estimation:

1. **Python package** combining Elastic Net penalties with coefficient bound constraints
2. **GCV framework** for hyperparameter selection without data splitting
3. **Simulation benchmarks** comparing PCReg-GCV against OLS across varying sample sizes
4. **Practical decision rules** for when to use constrained methods

## 0.2 Motivating Example

We demonstrate the problem using a learning curve dataset with only **5 training lots**—the practical minimum for learning curve analysis.

**Scenario Specifications:**

Table 1: Motivating example scenario parameters

| Parameter | Value |
|---|---|
| Training lots | 5 |
| Test lots | 12 |
| True $T_1$ | 100 |
| True Learning Rate | 90.0% |
| True Rate Effect | 80.0% |
| Predictor Correlation | 0.98 |
| CV Error | 0.2 |

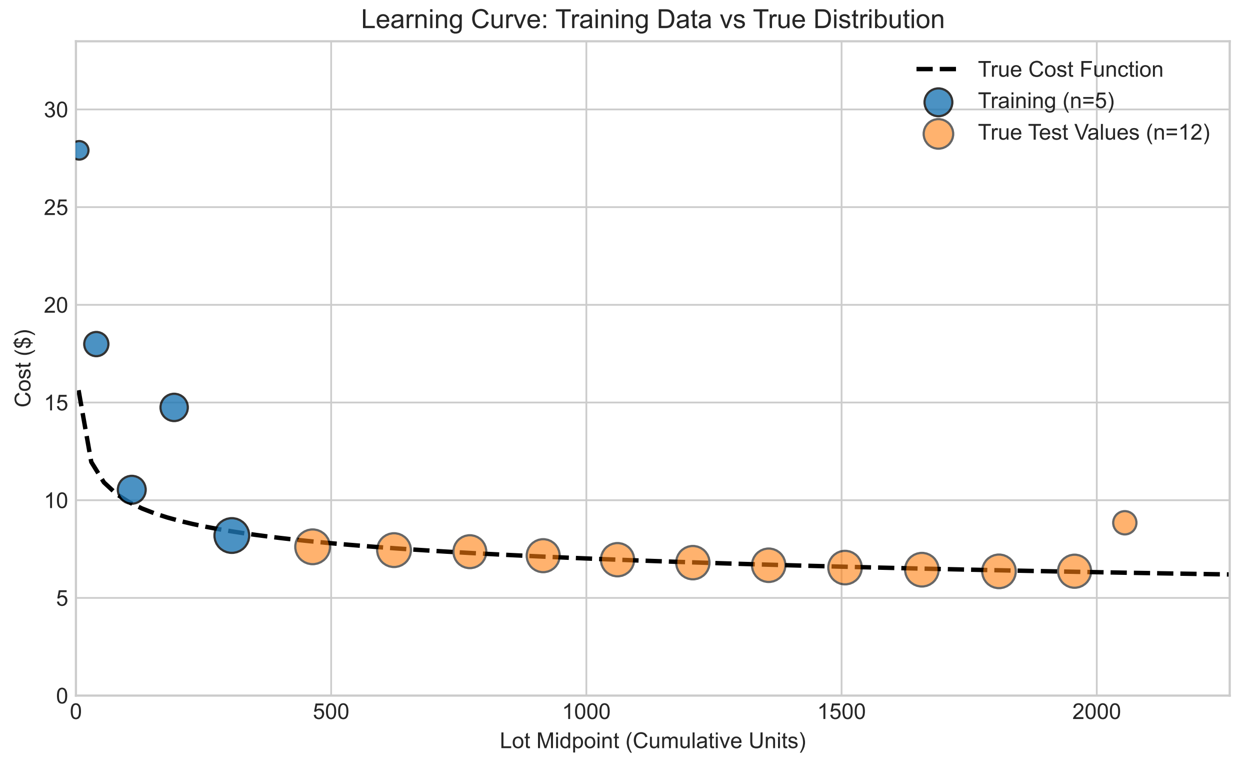

Figure 1: Learning curve data with 5 training lots (blue) and the true underlying cost distribution (dashed line). Point sizes reflect lot quantities. The goal is to predict the true relationship, not just fit the noisy observations.

### 0.2.1 Model Fitting

### 0.2.2 Results Comparison

Table 2: Comparison of estimation methods. PCReg-GCV has lower Train R² due to added bias from constraints, but better Test MAPE when predicting the true underlying relationship.

| Metric | True | OLS | OLS-LearnOnly | PCReg-GCV |
|---|---|---|---|---|
| $T_1$ | 100 | 210 | 48 | 70 |
| Learning Rate | 90.0% | 110.3% | 82.2% | 88.7% |
| Rate Effect | 80.0% | 57.6% | – | 86.3% |
| Valid Coefficients | Yes | NO | Yes | Yes |
| Train R² | – | 0.922 | 0.903 | 0.913 |
| Test MAPE (vs True) | – | 67.5% | 8.7% | 5.1% |

**Key Insight**: PCReg-GCV achieves a *lower* Train R² (0.913) than OLS (0.922). This is expected—constraints add bias to the training fit. However, this bias *improves* out-of-sample prediction, as shown by the lower Test MAPE against the true distribution.

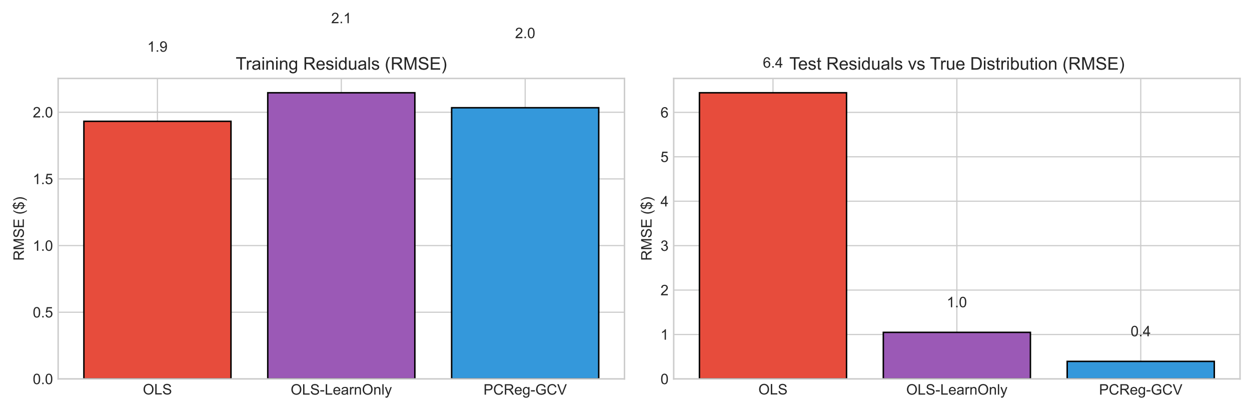

Figure 2: Residuals for each model on training data (left) and against true test values (right). PCReg-GCV shows larger training residuals but smaller errors when predicting the true underlying relationship.

### 0.2.3 Model Diagnostics and Bootstrap Analysis

The `penalized-constrained` package provides comprehensive diagnostics including bootstrap confidence intervals that compare constrained vs unconstrained estimation.

**Generalized Degrees of Freedom (GDF)**

For constrained models, computing proper degrees of freedom requires special consideration. When constraints are *binding* (coefficients at bounds), those parameters effectively lose freedom. Following Gaines et al. (2018):

$$\text{df} = |\text{Active predictors}| - |\text{Binding inequality constraints}|$$

For the PCReg-GCV model: GDF = 3.0 with 0 active constraint(s).

**Bootstrap Results Summary**

**Constrained Bootstrap** (with bounds and regularization):

- **T1**: Mean = 68.66, Std = 12.01, 95% CI = [44.25, 69.81]
- **b**: Mean = -0.1709, Std = 0.0562, 95% CI = [-0.2917, -0.0635]
- **c**: Mean = -0.2022, Std = 0.0645, 95% CI = [-0.3232, -0.0776]

**Unconstrained Bootstrap** (no bounds, alpha=0):

- **T1**: Mean = 196.60, Std = 197.29, 95% CI = [69.80, 673.64]
- **b**: Mean = 0.0163, Std = 0.3037, 95% CI = [-0.4129, 0.4683]
- **c**: Mean = -0.5556, Std = 0.4841, 95% CI = [-1.3528, 0.0324]

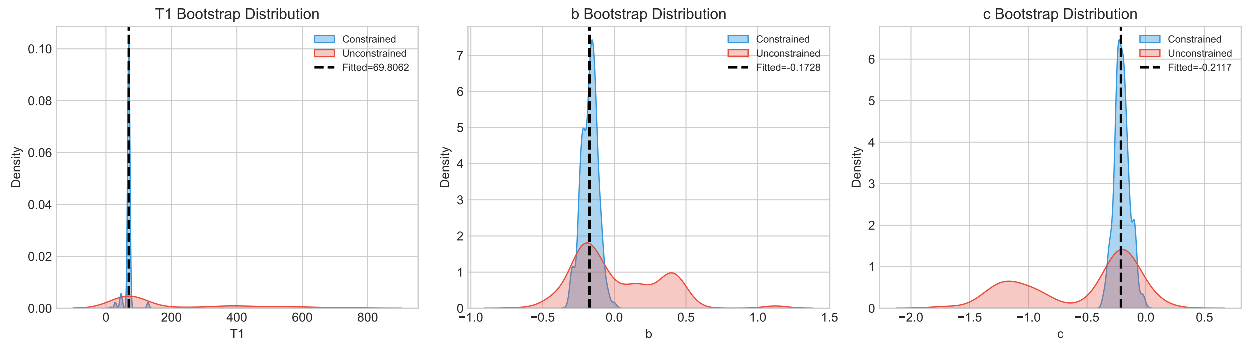

Figure 3: Bootstrap coefficient distributions comparing constrained (blue) vs unconstrained (red) estimation. Vertical lines show fitted values. Constraints reduce variance and keep estimates within economically plausible ranges.

**Key Diagnostic Insights:**

1. **Constraints at bounds**: When bootstrap samples frequently hit constraint boundaries, the data is "pulling" towards implausible values—exactly when constraints help most.
2. **Variance reduction**: Constrained bootstrap typically shows tighter distributions, reducing coefficient uncertainty at the cost of some bias.
3. **Divergence indicates constraint impact**: Large differences between constrained and unconstrained means show the constraints are actively shaping the solution.

**Note on nonlinear optimization**: PCReg uses numerical optimization (scipy's SLSQP), so classical regression assumptions don't directly apply. Bootstrap provides robust inference without these assumptions.

An interactive HTML diagnostic report with full bootstrap distributions has been saved to `output_v2/pcreg_diagnostic_report.html`.

## 0.3 Simulation Study

To systematically evaluate when PCReg-GCV outperforms OLS, we conducted a Monte Carlo simulation study with 8,100 scenarios.

### 0.3.1 Design

Table 3: Simulation study factorial design. 81 combinations × 100 replications = 8,100 scenarios.

| Factor | Levels |
|---|---|
| Sample size (n_lots) | 5, 10, 30 |
| CV error | 0.01, 0.1, 0.2 |
| Learning rate | 85%, 90%, 95% |
| Rate effect | 80%, 85%, 90% |

### 0.3.2 Data Generation

For each scenario, we:

1. **Randomly selected a quantity profile** from the SAR database (actual defense program procurement histories)
2. **Generated simulated average unit costs** using the learning curve model (Equation 1) with the scenario's true parameters
3. **Added multiplicative lognormal noise** with the specified coefficient of variation (CV)
4. **Split data**: First $n$ lots for training, remaining lots for test

This approach ensures realistic lot structures (varying quantities, realistic ramp-up patterns) while controlling the true underlying parameters. Note that the number of test lots varies by scenario depending on the program's total lot history—some programs have many available lots beyond training, others have few. This variability is acceptable as it reflects real-world conditions.

## 0.4 Key Findings

### 0.4.1 Finding 1: PCReg-GCV Outperforms OLS Regardless of Coefficient Reasonableness

A key finding is that PCReg-GCV provides better out-of-sample predictions **even when OLS produces reasonable coefficients**.

Table 4: PCReg-GCV win rates against OLS by coefficient reasonableness

| OLS Coefficients | N Scenarios | PCReg-GCV Win Rate (Test MAPE) |
|---|---:|---:|
| Reasonable (70-100%) | 6,745 | 33.7% |
| Unreasonable (<70% or >100%) | 1,355 | 78.7% |
| **Overall** | 8,100 | 41.2% |

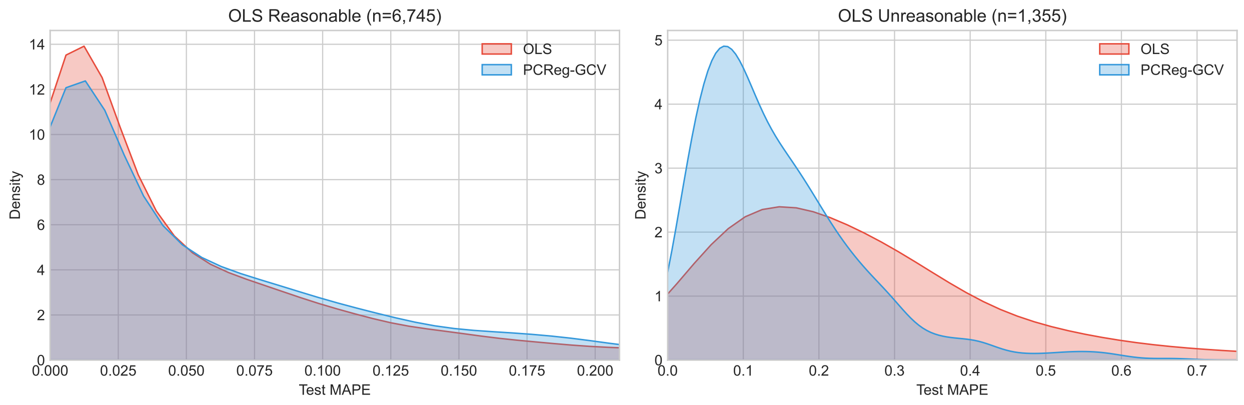

Figure 4: Distribution of Test MAPE for OLS vs PCReg-GCV, stratified by whether OLS produced reasonable coefficients. PCReg-GCV (blue) consistently shows lower MAPE in both cases.

### 0.4.2 Finding 2: OLS-LearnOnly Performs Poorly Outside Training Range

OLS with only the learning variable (OLS-LearnOnly) ignores the rate effect, which leads to poor extrapolation:

Table 5: Average Test MAPE by model

| Model | Mean Test MAPE |
|---|---:|
| OLS-LearnOnly | 15.9% |
| OLS | 9.9% |
| PCReg-GCV | 7.7% |

OLS-LearnOnly has worse Test MAPE than full OLS in **63.9%** of scenarios. While simplifying the model may seem appealing, omitting the rate effect leads to systematic prediction errors outside the training range.

### 0.4.3 Finding 3: Coefficient Bias Analysis

Constraints introduce bias in coefficient estimates. We analyze whether this bias is systematic and how it affects prediction:

Table 6: Mean coefficient bias (Estimated - True). Values near 0 indicate unbiased estimation.

| Coefficient | OLS Bias | PCReg-GCV Bias |
|---|---|---|
| $T_1$ (Mean Error) | 1108.37 | -3.59 |
| $b$ (Mean Error) | -0.0002 | -0.0078 |
| $c$ (Mean Error) | -0.0023 | 0.0348 |

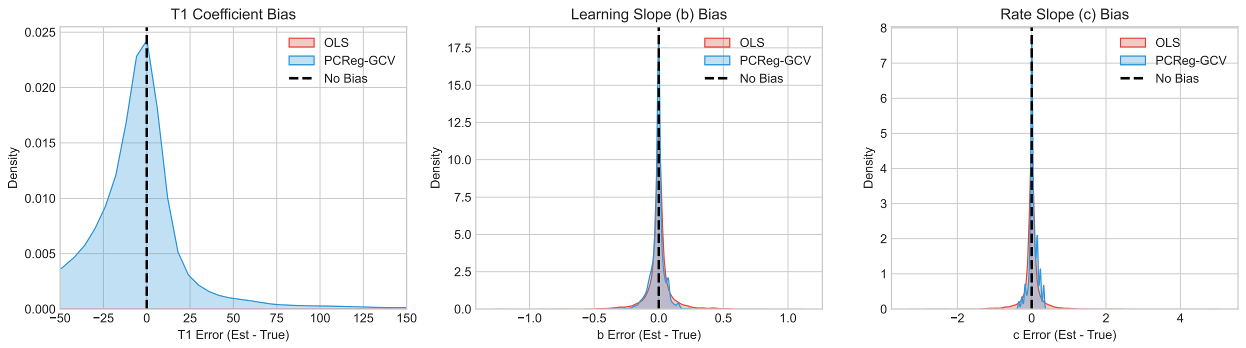

Figure 5: Distribution of coefficient errors by model. Vertical line at 0 indicates no bias. PCReg-GCV shows tighter distributions despite some bias, leading to lower variance in predictions.

**Key Insight**: While OLS is theoretically unbiased (mean errors near 0), it has high variance—the distribution is wide. PCReg-GCV may have slight bias but much lower variance, leading to better overall prediction accuracy (the classic bias-variance tradeoff).

### 0.4.4 Finding 4: Sample Size and Noise Effects

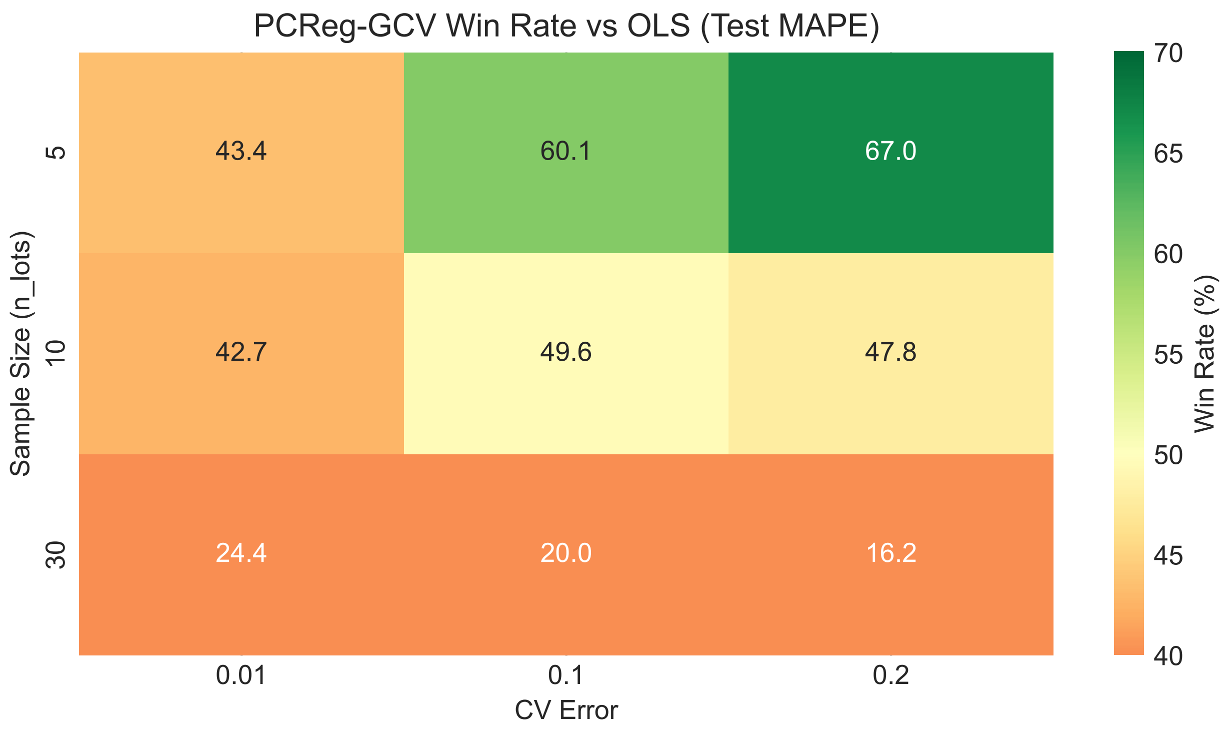

Figure 6: PCReg-GCV win rate by sample size and CV error. Green indicates PCReg-GCV advantage (>50%).

## 0.5 Recommendations: When to Use PCReg-GCV

We trained a decision tree to identify conditions where PCReg-GCV is most beneficial, using only features available to practitioners (not true parameter values):

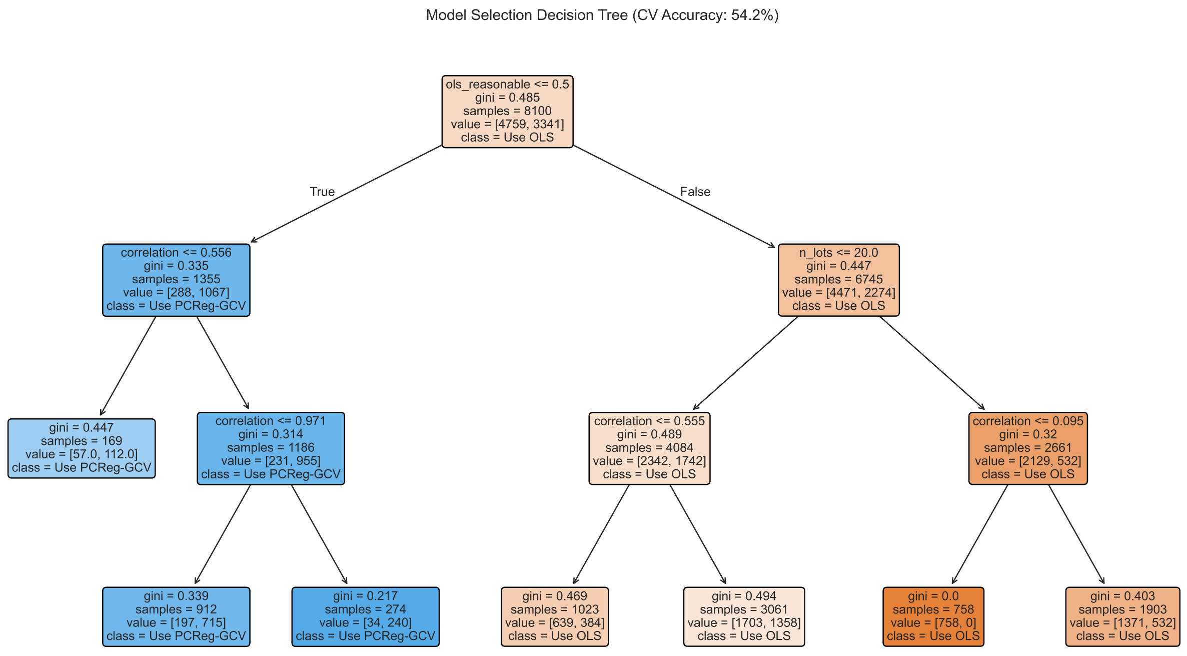

Figure 7: Decision tree for model selection. Green nodes favor PCReg-GCV; orange nodes favor OLS. The tree uses only information available to the analyst.

**Practical Decision Rules:**

Based on the decision tree and simulation analysis:

1. **If OLS produces unreasonable coefficients** (LC or RC outside 70-100%): **Use PCReg-GCV**
2. **If sample size is 10 lots or fewer**: **Prefer PCReg-GCV** (more stable)
3. **If predictor correlation > 0.9**: **Prefer PCReg-GCV** (multicollinearity protection)
4. **Otherwise**: Either method acceptable, but PCReg-GCV still slightly favored

### 0.5.1 Software

The `penalized-constrained` Python package was developed specifically for the cost estimating community:

```
pip install penalized-constrained
```

## 0.6 Conclusions

1. **PCReg-GCV improves out-of-sample prediction** even when OLS produces reasonable coefficients
2. **Constraints introduce beneficial bias** that reduces prediction variance
3. **OLS-LearnOnly performs poorly** outside the training range—don't oversimplify
4. **GCV enables reliable hyperparameter selection** with as few as 5 training observations
5. **Simple decision rule**: When in doubt, use PCReg-GCV; the downside is minimal

## References

Flynn, Brian, and Andy James. 2016. "Multicollinearity in CER Development." In *ICEAA Professional Development & Training Workshop*.

James, Gareth M., Courtney Paulson, and Paat Rusmevichientong. 2020. "Penalized and Constrained Optimization: An Application to High-Dimensional Website Advertising." *Journal of the American Statistical Association* 115 (529): 107–22. https://doi.org/10.1080/01621459.2019.1609970.

# A Appendix A: Simulation Details

## A.1 Data Generation Process

For each of the 8,100 scenarios:

1. **Select quantity profile**: Randomly sample a defense program from the SAR database with sufficient lot history
2. **Extract lot structure**: Use actual procurement quantities and calculate lot midpoints
3. **Generate true costs**: Apply learning curve model with scenario parameters
4. **Add noise**: Multiply true costs by lognormal error: $Y_{obs} = Y_{true} \cdot e^{\epsilon}$ where $\epsilon \sim N(-\sigma^2/2, \sigma^2)$
5. **Split data**: First $n$ lots for training, **all remaining lots** for test (variable test set size)

## A.2 Model Specifications

Table 7: Models compared in main paper

| Model | Description | Constraints |
|---|---|---|
| OLS | Standard log-log OLS | No |
| OLS_LearnOnly | OLS with learning variable only | No |
| PCReg_GCV | GCV-selected penalty + constraints | Yes |

## B Appendix B: Full Results (All Models)

Table 8: Overall model performance across all 8,100 scenarios. Lower Test MAPE is better.

| Model | Test MAPE | Test SSPE | b Error | c Error | R2 |
|---|---:|---:|---:|---:|---:|
| PCReg_GCV_Tight | 0.0561 | 0.0861 | 0.0200 | 0.0337 | 0.852 |
| PCReg_ConstrainOnly | 0.0764 | 0.2052 | 0.0354 | 0.0791 | 0.877 |
| PCReg_GCV | 0.0773 | 0.2074 | 0.0338 | 0.0827 | 0.871 |
| PCRegGCV_LogMSE | 0.0778 | 0.2832 | 0.0341 | 0.0776 | 0.877 |
| PCReg_CV | 0.0794 | 0.2538 | 0.0353 | 0.0814 | 0.862 |
| BayesianRidge | 0.0801 | 0.4595 | 0.0360 | 0.0886 | 0.882 |
| PCReg_AICc | 0.0808 | 0.2171 | 0.0349 | 0.0913 | 0.863 |
| RidgeCV | 0.0952 | 0.9362 | 0.0481 | 0.1216 | 0.896 |
| LassoCV | 0.0957 | 0.9767 | 0.0428 | 0.1082 | 0.858 |
| OLS | 0.0994 | 0.9727 | 0.0520 | 0.1319 | 0.897 |
| OLS_LearnOnly | 0.1592 | 0.9543 | 0.0827 | 0.2361 | 0.789 |

## C Appendix C: Software Documentation

### C.1 Installation

```
pip install penalized-constrained
```

### C.2 Basic Usage

```python
import penalized_constrained as pcreg
import numpy as np

model = pcreg.PenalizedConstrainedCV(
    coef_names=['T1', 'b', 'c'],
    bounds={
        'T1': (0, None),      # T1 must be positive
        'b': (-0.5, 0),       # Learning rate 70-100%
        'c': (-0.5, 0)        # Rate effect 70-100%
    },
    prediction_fn=lambda X, p: p[0] * X[:,0]**p[1] * X[:,1]**p[2],
    loss='sspe',
    selection='gcv'
)
model.fit(X_train, y_train)
```

### C.3 Citation

```
@inproceedings{joy2026pcreg,
  title={Penalized-Constrained Regression: Combining Regularization
         and Domain Constraints for Cost Estimation},
  author={Joy, Kevin and Watstein, Max},
  booktitle={ICEAA Professional Development \& Training Workshop},
  year={2026}
}
```