Penalized-Constrained Regression

Combining Regularization and Domain Constraints for Cost Estimation

Kevin Joy   Max Watstein

2026-02-11

Table of contents

Abstract

Small datasets with intercorrelation pose serious challenges to the stability of coefficients generated by Ordinary Least Squares (OLS) regression. A motivating example in cost estimating is Learning Curve with Rate Effect analysis, where datasets are typically small, the lot midpoint (Learning) is correlated to lot quantity (Rate) as production ramps up, and slopes are expected to be $\leq 100\%$. Lasso, Ridge, and Elastic Net regularization methods address multicollinearity by penalizing coefficients. Separately, constrained optimization methods can impose explicit restrictions on coefficient values when prior knowledge about their behavior is known—such as bounding slopes within a known range. This paper investigates the combined effects of penalized regularization methods and constrained optimization. We explore how to assess model stability and goodness-of-fit using likelihood-free diagnostic techniques suited to optimization-based regressions, such as cross-validation for generalization error.

1 Introduction

While developing Cost Estimating Relationships (CERs) for small datasets (5-30 data points), a recurring pattern emerged. The regression models often showed strong fit statistics with high R^2 and low standard error, but nonsensical coefficients. Coefficients often had wrong signs, implausible magnitudes, and poor p-values. As Department of Defense (DoD) analysts, this story may feel all too familiar.

Intercorrelated datasets are a frequent presence in cost analysis, causing regression models to misbehave. This is especially true for Learning Curve with Rate Effect analysis, where datasets are typically small and the lot midpoint is correlated to lot quantity. Traditional remedies for multicollinearity can be insufficient for learning curve datasets. Increasing sample size is infeasible as it requires waiting years for additional production lots and cost data to become available. Dropping explanatory variables, such as rate effect, leads to model misspecification that ignores a fundamental cost driver. This issue is compounded when predicting outside the relevant range which is very common. Even when these remedies are employed, OLS can still produce implausible coefficients with the wrong signs, such as learning curve slopes >100%. Regularization techniques like Ridge, Lasso, and Elastic Net can help stabilize estimates, but they don't guarantee plausible coefficients. Constrained optimization can enforce domain knowledge, but without regularization, it may not solve the underlying multicollinearity problem. This paper explores the combination of penalized regularization and constrained optimization to address both challenges.

For our research, we reviewed existing literature on multicollinearity, regularization, constrained optimization, and learning theory. Drawing on these foundations, we developed Penalized-Constrained Regression (PCReg), which integrates Elastic Net penalties with domain-knowledge coefficient constraints. We then used Monte Carlo simulation to evaluate PCReg's accuracy estimating the true parameters and compared it's performance against OLS across a range of sample sizes and collinearity conditions, measuring both coefficient recovery and out-of-sample prediction accuracy.

This paper develops and validates Penalized-Constrained Regression for cost estimation, providing:

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 against OLS across varying sample sizes
4. **Practical decision rules** for when to use constrained methods

2 Motivating Example

2.1 The Learning Curve Problem

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

$$\text{Average Unit Cost} = T_1 \cdot (\text{Lot Midpoint})^b \cdot (\text{Lot Quantity})^c \cdot \varepsilon$$

where:

- T_1 = theoretical first unit cost
- b = learning slope in log space
- c = rate slope in log space
- ε = multiplicative error

b and c are both slopes in log space. When transformed to unit space, they become the learning curve slope (LCS $= 2^b$) and the rate effect (2^c), both typically ranging from 70-100%. As DoD programs ramp up production from Engineering Manufacturing Development (EMD) to Full Rate Production (FRP), lot midpoint (the learning predictor) becomes highly correlated with lot quantity (the rate predictor). By definition LCS and Rate Effect are $\leq$ 100%, meaning costs decrease, not increase, with cumulative production or lot quantity.

If estimated LCS or Rate Effect are >100% this is often an indicator that other explanatory variables such as supply chain disruptions or diseconomies of scale are missing and need to be modeled, not a violation of the definition. Once these factors are properly modeled as separate explanatory variables, estimated LCS and Rate Effect should be $\leq$ 100%, consistent with their definition.

2.2 Example Scenario

We demonstrate the "Learning Curve Problem" using the following dataset which was generated by our Simulation Study ***insert cross reference**.

Table 1: Motivating example training dataset.

lot_midpoint	lot_quantity	observed_cost
11.694	30.000	44.822
44.592	30.000	43.961
79.598	40.000	42.172
129.251	60.000	39.040
226.613	140.000	38.229
414.709	241.000	26.349
713.347	360.000	22.446
1076.104	360.000	21.463
1437.463	360.000	24.394
1812.743	390.000	24.922

Using OLS, we get the following results:

Metric	OLS
T_1	114
Learning Rate	100.8%
Rate Effect	82.5%

We see find that by using OLS this example dataset yields invalid coefficents, specifcally a LCS of 100.8% which violates our definiton of LCS being $\leq$ 100%.

3 Method

3.1 Objective Function

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

$$\text{Objective:} \quad \arg\min_{\theta} \underbrace{\sum_{i=1}^{n} L(y_i, \hat{y}_i)}_{\text{Loss}} + \underbrace{\lambda\Big[\alpha\, L_1 + \frac{1-\alpha}{2}\, L_2\Big]}_{\text{Elastic Net Penalty}}$$

$$\text{Subject To:} \quad \theta_{\text{lower}} \leq \theta \leq \theta_{\text{upper}} \quad \text{(Componentwise)}$$

where:

- $L(y_i, \hat{y}_i)$ = Loss function measuring prediction error
- $\theta_{\text{lower}}, \theta_{\text{upper}}$ = Componentwise bounds on coefficients
- λ = Regularization parameter
- L_1 = Lasso penalty term
- L_2 = Ridge penalty term
- α = Mixing parameter, determining balance between L1 vs L2

The lower and upper bounds (θ_{lower} and θ_{upper}) can be set to achieve loose bounds or tight bounds for the coefficient:

- **Loose bounds** (e.g., 70-100%): Wide range which allows flexibility while preventing egregious coefficient violations
- **Tight bounds** (e.g., 85-95%): Narrower range which incorporates strong prior knowledge but risks over-constraining coefficients

Our research builds on the contributions of many others. *** CITE RIDGE PAPER *** established Classical Ridge regression theory and proves that there exists a λ penalty that minimizes mean squared error. More recently James, Paulson, and Rusmevichientong (2020) demonstrated that constrained Lasso can achieve optimal prediction when coefficients are constrained (positively or negatively).

3.2 PCReg

3.2.1 Scikit-learn

3.2.2 GCV

***Need some discussion of GCV**

3.2.3 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):

$$\text{GDF} = n - p - |\text{Binding inequality constraints}|$$

where n is the sample size and p is the number of active predictors (non-zero coefficients).

For the PCReg model: GDF = 7.0 with 0 active constraint(s).

Note on Hu's Alternative Formula: Hu (2010) proposes a different approach where *all specified constraints* count against degrees of freedom, not just binding ones: GDF $= n - p - (\#\text{ Constraints}) + (\#\text{ Redundancies})$. Hu's method is more conservative, always reducing GDF when constraints are specified, while Gaines' method only penalizes constraints that are actually active at the solution. The `penalized-constrained` package supports both methods via the `gdf_method` parameter.

3.3 Model Diagnostics

List Model Diagnostics

3.4 Bootstrap Analysis

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

Bootstrap Results Summary

Constrained Bootstrap (with bounds and regularization):

- **T1**: Mean = 93.98, Std = 10.41, 95% CI = [68.63, 100.00]
- **b**: Mean = -0.0658, Std = 0.0425, 95% CI = [-0.1806, -0.0110]
- **c**: Mean = -0.1497, Std = 0.0597, 95% CI = [-0.2275, -0.0000]

Unconstrained Bootstrap (no bounds, alpha=0):

- **T1**: Mean = 109.20, Std = 14.03, 95% CI = [99.99, 140.81]
- **b**: Mean = -0.0059, Std = 0.0704, 95% CI = [-0.1781, 0.1105]
- **c**: Mean = -0.2493, Std = 0.0903, 95% CI = [-0.3988, -0.0545]

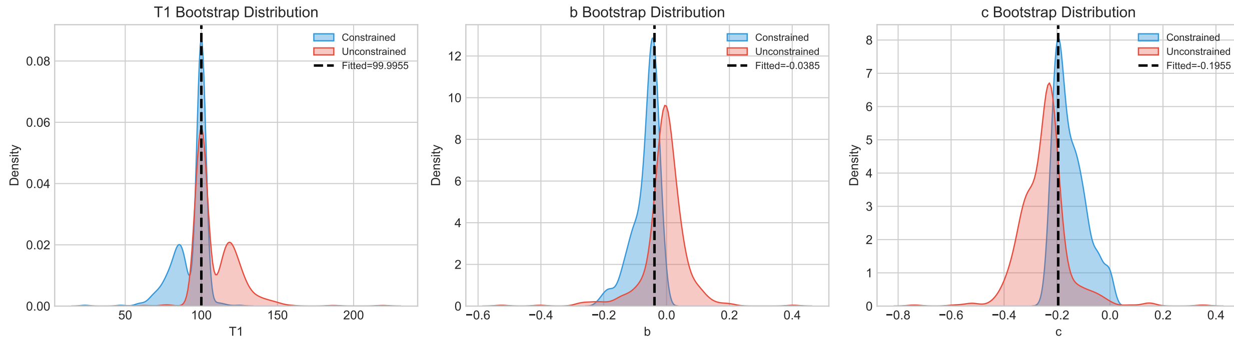

Figure 1: Bootstrap coefficient distributions comparing constrained (blue) vs unconstrained (red) estimation. Vertical lines show fitted values. Constraints reduce variance and keep estimates within economically plausible ranges.

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`.

4 Simulation Study

We performed a Monte Carlo simulation study evaluate PCReg and compare with traditional OLS.

4.1 Siumulation 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%

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 (**?@eq-learning-curve**) 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 or none. This variability is acceptable as it reflects real-world conditions.

4.2 Example Scenario

We return to the example scenario which was introduce in ***insert cross reference***. Below we see the scenario specifications which generated our example scenario dataset.

Table 4: Example scenario parameters

Parameter	Value
Training lots	10
Test lots	20
True T_1	100
True Learning Rate	95.0%
True Rate Effect	90.0%
Predictor Correlation	0.95
CV Error	0.1

Here is the example dataset with the training data from ***insert crossreferece** highlighted green:

Table 5: Motivating example dataset with training and test lots.

lot_type	lot_midpoint	lot_quantity	observed_cost	true_cost
train	11.694	30	44.822	49.71
train	44.592	30	43.961	45.022
train	79.598	40	42.172	41.287
train	129.251	60	39.040	37.451
train	226.613	140	38.229	31.586
train	414.709	241	26.349	27.811
train	713.347	360	22.446	25.136
train	1076.104	360	21.463	24.382
train	1437.463	360	24.394	23.865
train	1812.743	390	24.922	23.176
test	2208.256	400	21.974	22.752
test	2669.377	525	18.962	21.526
test	3231.513	600	19.925	20.798
test	3789.404	512	20.783	21.056
test	4320.366	550	17.697	20.627
test	4866.798	542	20.660	20.491
test	5408.087	540	21.409	20.343
test	5955.193	554	19.309	20.12
test	6509.390	554	17.064	19.988
test	7063.555	554	20.573	19.868
test	7615.722	550	19.931	19.779
test	8165.842	550	19.317	19.677
test	8715.947	550	22.181	19.583
test	9266.039	550	18.682	19.494
test	9816.121	550	21.034	19.411
test	10366.194	550	20.868	19.333
test	10916.260	550	18.288	19.259
test	11466.319	550	18.351	19.189
test	12016.373	550	18.547	19.123
test	12556.499	530	16.691	19.168

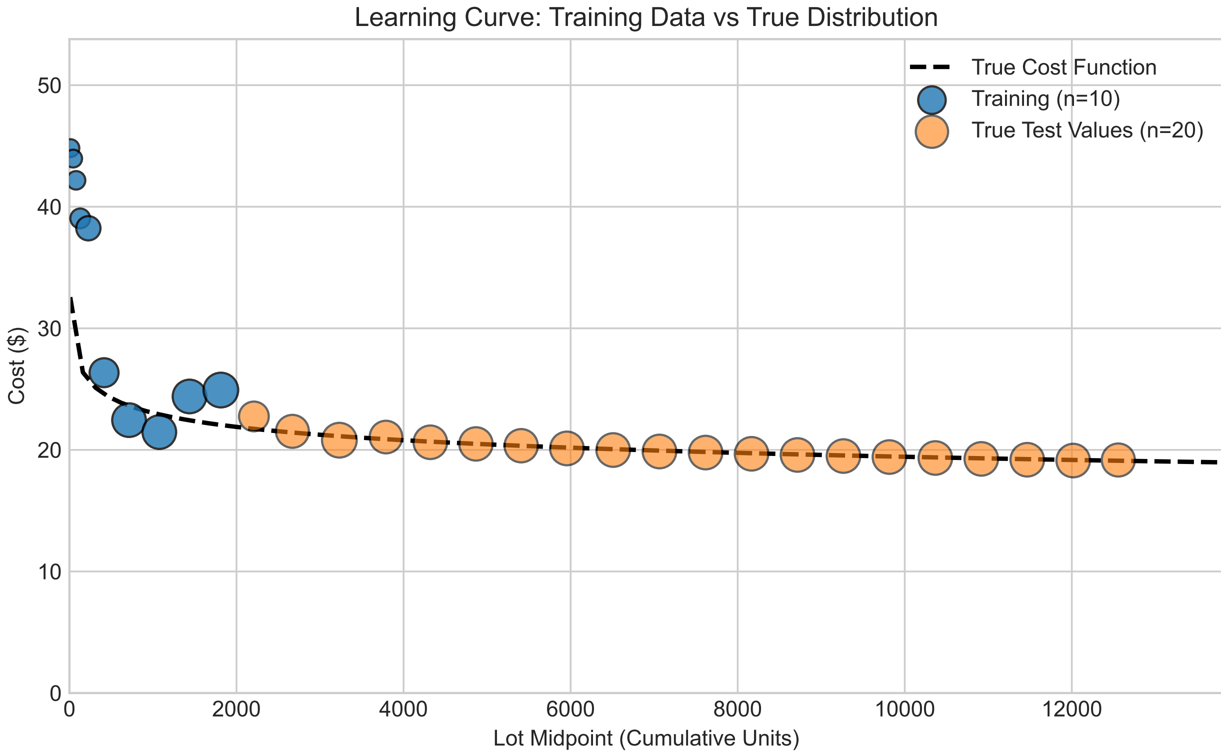

Figure 2: Learning curve data with `python n_train` training lots (blue) and the true underlying cost distribution (dashed line). Point sizes reflect lot quantities. The goal is to predict the true relationship, not just fit the noisy observations.

Table 6: Comparison of estimation methods. PCReg has lower Train R^2 due to added bias from constraints, but better Test MAPE when predicting the true underlying relationship.

Metric	True	OLS	OLS-LearnOnly	PCReg
T_1	100	114	79	100
Learning Rate	95.0%	100.8%	89.2%	97.4%
Rate Effect	90.0%	82.5%	–	87.3%
Valid Coefficients	Yes	NO	Yes	Yes
Train R^2	–	0.912	0.796	0.907
Test MAPE (vs True)	–	9.8%	8.4%	3.9%

Key Insight: OLS achieves a *lower* Train R^2 OLS (0.912) than PCReg (0.907). This is expected, penalties and 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.

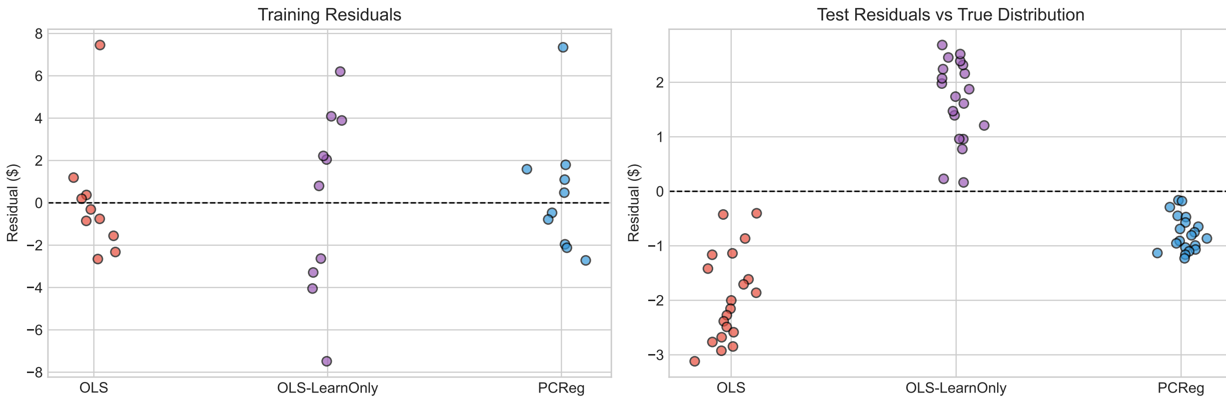

Figure 3: Residuals for each model on training data (left) and against true test values (right). PCReg shows larger training residuals but smaller errors when predicting the true underlying relationship.

5 Key Findings

Across 8,100 simulation scenarios, OLS produced economically unreasonable coefficients (learning curve or rate effect outside 70-100%) in **16.7%** of cases (1,355 scenarios).

5.1 Finding 1a: PCReg Significantly Outperforms OLS When Coefficients Are Unreasonable

When OLS produces unreasonable coefficients, PCReg wins **78.7%** of scenarios on Test MAPE.

Metric	OLS Mean	PCReg Mean	OLS Median	PCReg Median
Test MAPE	0.2862	0.1456	0.2094	0.1179
T1 APE	66.5658	0.5108	0.5694	0.3418
LC Abs Error	0.1182	0.0483	0.0943	0.0489
RC Abs Error	0.3548	0.1219	0.2347	0.1

: Performance comparison when OLS produces unreasonable coefficients (n=1,355). Lower values are better. {#tbl-unreasonable}

Statistical Significance: Wilcoxon signed-rank test confirms PCReg significantly outperforms OLS on Test MAPE (p < nan).

5.2 Finding 1b: PCReg Performs Comparably When OLS Coefficients Are Reasonable

When OLS produces reasonable coefficients, OLS "wins" on Test MAPE in **66.3%** of scenarios. However, the performance differences are negligible in practical terms.

Metric	OLS Mean	PCReg Mean	OLS Median	PCReg Median
Test MAPE	0.0549	0.061	0.0253	0.0298
T1 APE	0.1628	0.1562	0.0623	0.0709
LC Abs Error	0.0156	0.0154	0.007	0.007
RC Abs Error	0.0304	0.0353	0.0153	0.0172

: Performance comparison when OLS produces reasonable coefficients (n=6,745). {#tbl-reasonable}

Key Insight: When OLS coefficients are reasonable, the mean Test MAPE difference is only **0.61 percentage points** (11.1% relative difference). While statistically detectable (Wilcoxon p=nan), this difference is negligible in practice. OLS's occasional large errors (visible in the heavy right tail) inflate the mean, but the median performance is nearly identical.

5.3 Summary: Win Rates by Coefficient Reasonableness

Table 9: PCReg win rates against OLS by coefficient reasonableness

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

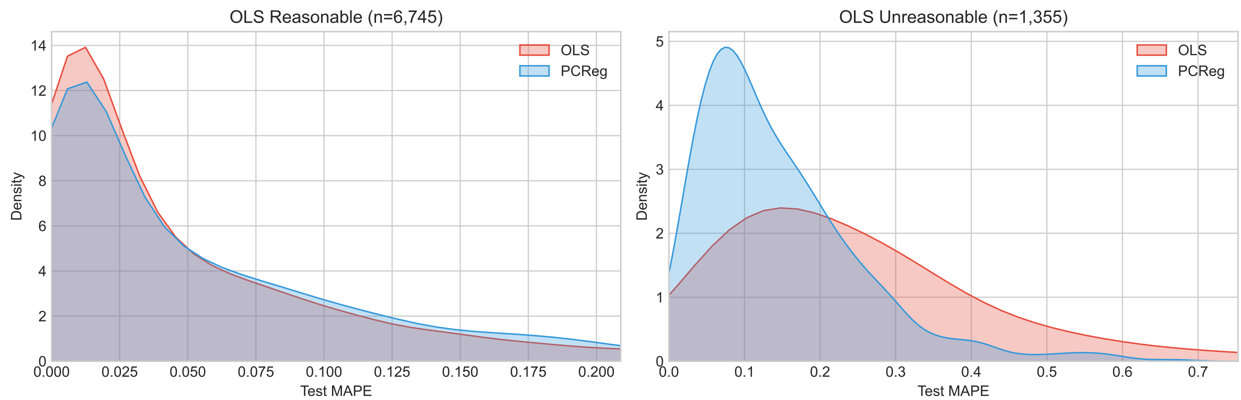

Figure 4: Distribution of Test MAPE for OLS vs PCReg, stratified by whether OLS produced reasonable coefficients. When unreasonable (right), OLS shows a heavy right tail of large errors that PCReg avoids.

5.4 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 10: Average Test MAPE by model

Model	Mean Test MAPE
OLS-LearnOnly	15.9%
OLS	9.9%
PCReg	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.

PCReg outperforms OLS-LearnOnly on Test MAPE in **67.9%** of all scenarios:

- When OLS coefficients are **reasonable**: PCReg wins **70.9%**
- When OLS coefficients are **unreasonable**: PCReg wins **52.8%**

Across all scenarios, PCReg outperforms OLS on Test MAPE in **41.2%** of cases, indicating a consistent predictive advantage even when OLS is competitive.

5.4.1 Finding 3: Coefficient Bias Analysis

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

Table 11: Coefficient bias statistics (Estimated - True). Mean/Median near 0 indicates unbiased estimation; lower Std indicates more stable estimates.

	Coefficient	Statistic	OLS	PCReg
0	T_1	Mean	1108.37	-3.58
1	T_1	Median	-0.22	-2.73
2	T_1	Std	69135.60	46.12
3	b	Mean	-0.0002	-0.0078
4	b	Median	-0.0002	-0.0017
5	b	Std	0.11	0.05
6	c	Mean	-0.0023	0.03
7	c	Median	0.0003	0.01
8	c	Std	0.29	0.12

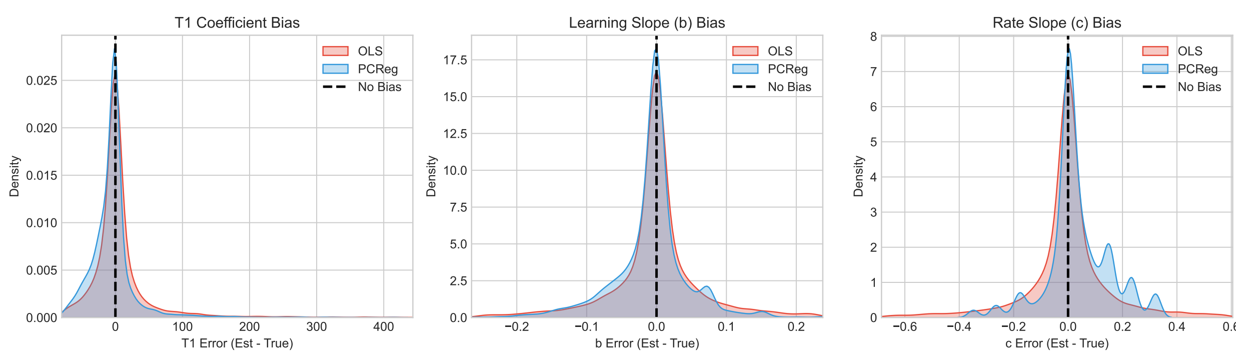

Figure 5: Distribution of coefficient errors by model (1st-99th percentile). Vertical line at 0 indicates no bias. PCReg shows tighter distributions despite some bias, leading to lower variance in predictions.

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

5.5 Finding 4: Parameter Effects on Model Performance

Beyond coefficient reasonableness, several design factors systematically influence when PCReg outperforms OLS. We examine sample size, predictor correlation, and noise level.

By OLS Coefficient Reasonableness:

OLS Coefficients	N	PCReg Win Rate	OLS Mean MAPE	PCReg Mean MAPE
Unreasonable	1355	78.7%	28.6%	14.6%

OLS Coefficients	N	PCReg Win Rate	OLS Mean MAPE	PCReg Mean MAPE
Reasonable (70-100%)	6745	33.7%	5.5%	6.1%

By Sample Size (n_lots):

Sample Size	N	PCReg Win Rate	% OLS Unreasonable	OLS Mean MAPE	PCReg Mean MAPE
5	2700	56.9%	32.0%	16.2%	10.7%
10	2700	46.5%	16.8%	7.4%	6.4%
30	2700	20.2%	1.4%	3.7%	4.8%

By Predictor Correlation:

Correlation	N	PCReg Win Rate	% OLS Unreasonable	OLS Mean MAPE	PCReg Mean MAPE
<0.80	3187	31.8%	8.5%	6.7%	6.5%
0.80-0.90	873	52.9%	22.1%	10.3%	7.7%
0.90-0.95	2204	38.9%	15.2%	9.8%	7.6%
>0.95	1416	56.5%	35.0%	14.5%	8.7%

By Noise Level (CV Error):

CV Error	N	PCReg Win Rate	% OLS Unreasonable	OLS Mean MAPE	PCReg Mean MAPE
0.01	2700	36.7%	0.1%	0.9%	1.0%
0.1	2700	43.3%	16.2%	9.4%	8.1%
0.2	2700	43.7%	33.9%	19.6%	14.1%

Key Observations:

1. **Sample Size Effect**: As sample size decreases, OLS becomes less stable. With n=5 lots, PCReg wins **56.9%** of scenarios compared to **20.2%** at n=30. The rate of unreasonable OLS coefficients also increases from **1.4%** to **32.0%** as sample size decreases.
2. **Correlation Effect**: Higher predictor correlation destabilizes OLS. At correlation >0.95, PCReg wins **56.5%** of scenarios versus **31.8%** at lower correlations. This is the multicollinearity effect. When predictors are highly correlated, OLS coefficient estimates become unstable and constraints provide crucial stability.
3. **Noise Level Effect**: Higher CV error increases the advantage of PCReg. With more noise, OLS has greater difficulty separating learning and rate effects, leading to more unreasonable coefficient estimates.

4. **Compounding Effects**: These factors interact—small samples with high correlation and high noise represent the most challenging scenarios for OLS, where PCReg provides the greatest benefit.

6 Recommendations: When to Use PCReg

Practical Decision Rules:

1. **If OLS produces unreasonable coefficients** (LC or RC outside 70-100%): **Use PCReg**, it significantly outperforms OLS in these scenarios
2. **For small samples (n 5 or 10 lots) with noisy data**: **Prefer PCReg**, it wins 60-67% of scenarios
3. **For large samples (n 30 lots)**: **OLS is generally preferred**, it wins ~80% of scenarios
4. **For intermediate cases**: Either method is acceptable; PCReg provides insurance against unreasonable coefficients with minimal downside

Bottom line: The difference between OLS and PCReg is typically small and can be controlled by explicitly defining loose constraints and arbitrarily small penalties.

6.1 Software

The `penalized-constrained` Python package was developed specifically for the cost estimating community. As of this paper's publication, the software is in active development but has achieved stable functionality.

Installation: While we anticipate release to PyPI for convenient installation via `pip install penalized-constrained`, the package is currently available via GitHub. To install from the development repository:

```
pip install git+https://github.com/frankij11/Penalized-Constrained-Regression.git
```

For quick-start guides and basic usage examples, see the package documentation on GitHub or Section C in the appendices.

A key advantage of this framework is that **PCReg collapses to OLS** when no constraints are defined, no penalties are applied (=0), and a linear functional form is used. This allows analysts to use a single, cohesive library for all regression analysis. From standard OLS to fully constrained penalized models with custom prediction functions all without switching tools or workflows. This library is designed to integrate seamlessly Python's scikit-learn data science workflows and allow machine learning techniques to solve for optimal parameters.

6.2 Summary

Our simulation study demonstrates that PCReg provides meaningful advantages in small-sample, high-noise scenarios where OLS is most likely to produce unreasonable coefficients, while large samples favor standard OLS. The constraints introduce beneficial bias that reduces prediction variance—a classic bias-variance tradeoff that works in the analyst's favor when data are limited. With GCV enabling reliable hyperparameter selection using as few as 5 observations and OLS-LearnOnly's poor extrapolation performance reinforcing that omitting the rate effect oversimplifies the problem, analysts have clear guidance: match the method to the sample size and data quality, with PCReg serving as effective insurance when uncertainty is high.

References

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 16: 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 17: 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}
}
```