

Regression & Prediction

Theory and Practice with House Prices

Your Name

February 23, 2025

xAI

Our Housing Journey

- Starting Simple:
Foundations
- Expanding the Scope:
Complexity
- Refining Precision:
Optimization
- Facing Challenges: Pitfalls
- Insights & Next Steps:
Applications

Focus

Unpacking King County house prices with rich theory and dual R/Python implementations

Starting Simple

The Housing Puzzle

- **Objective:** Decode drivers of house prices in King County—size, location, features
- **Regression:** A statistical lens linking price (Y) to predictors like size (X)
- **Purpose:** Explain historical sales patterns and forecast future values for buyers and assessors

Simple Linear Regression: Theory & R

- **Theory (p. 141):** Models a straight-line relationship:
$$Y = b_0 + b_1X + e$$
- b_0 : Base price when size is zero;
 b_1 : Price increase per sq ft; e : Random error
- Assumes linearity and independence—foundation of regression

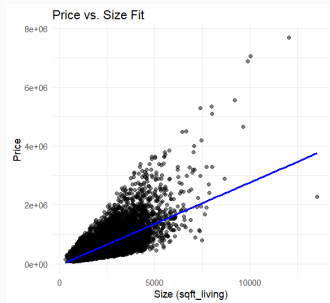


Figure 1: Price vs. Size Fit

```
1 # R (p. 152 adapted)
2 simple_lm <- lm(AdjSalePrice ~ SqFtTotLiving, data
  = house)
3 # Output: b_0 ~ base, b_1 ~ price/sq ft
```

Simple Linear Regression: Python

- **Practice:** Fits price to living space, revealing size's impact
- **Key Insight:** Positive slope shows larger homes fetch higher prices

- Size as a core driver

```
1 # Python (p. 152 adapted)
2 from sklearn.linear_model import LinearRegression
3 predictors = ['SqFtTotLiving']
4 outcome = 'AdjSalePrice'
5 simple_lm = LinearRegression()
6 simple_lm.fit(house[predictors], house[outcome])
7 # Example: Intercept ~ base, Coef ~ $ per sq ft
```

Finding the Best Fit: Least Squares

- **How do we draw that line?** We minimize the mess—sum of squared errors
- **Theory:** Finds the line minimizing residual sum of squares:
$$\sum_{i=1}^n (Y_i - \hat{Y}_i)^2$$
- **How:** Adjusts b_0 and b_1 to reduce prediction errors—optimal for linear fits
- **History:** Legendre (1805) and Gauss; computationally efficient but outlier-sensitive in small datasets

Expanding the Scope

More Clues: Multiple Linear Regression

- **Theory:** Extends to multiple predictors:
$$Y = b_0 + b_1X_1 + b_2X_2 + \cdots + e$$
- **Power:** Captures combined effects—size, lot, bedrooms—assuming linearity
- **Use:** Explains complex housing dynamics

- Size adds \$229/sq ft

```
1 # R (p. 152)
2 house_lm <- lm(AdjSalePrice ~ SqFtTotLiving +
3   SqFtLot + Bathrooms +
4   Bedrooms + BldgGrade, data = house)
# Coefs: SqFtTotLiving = 228.831, Bedrooms =
-47769.955
```

Multiple Linear Regression: Python

- **Practice:** Models housing with multiple factors
- **Key Insight:** Negative bedroom coef suggests smaller rooms hurt value

- Bedrooms vs. size tension

```
1 # Python (p. 152)
2 predictors = ['SqFtTotLiving', 'SqFtLot', '
    Bathrooms', 'Bedrooms', 'BldgGrade']
3 house_lm = LinearRegression()
4 house_lm.fit(house[predictors], house[outcome])
5 # Intercept: -521871.168, Bedrooms: -47769.955
```

Factor Variables: Theory & R

- **Theory (p. 163):** Encodes categorical variables (e.g., property type) as binary dummies
- **Why:** Allows non-numeric factors in regression; compares to a reference level
- **Example:** Single-family vs. Townhouse effects
- Type shifts price

```
1 # R (p. 164)
2 prop_type_dummies <- model.matrix(~ PropertyType -
3   1, data = house)
# Output: 1 for each type present
```

Factor Variables: Python

- **Practice:** Integrates property type into price model
- **Key Insight:** Townhouses may differ from single-family homes

- Baseline comparison

```
1 # Python (p. 166 adapted)
2 import pandas as pd
3 X = pd.get_dummies(house['PropertyType'], drop_
    first=True)
4 # Drops first level (e.g., Multiplex) as reference
```

Nonlinear Fit: Theory & Splines in R

- **Theory (p. 187):** Nonlinear via polynomial segments joined at knots
- **Why:** Captures diminishing returns—e.g., small homes gain more per sq ft
- **Advantage:** Flexible fit without overfitting like high-order polynomials

```
1 # R (p. 190)
2 library(splines)
3 knots <- quantile(house_98105$SqFtTotLiving, p = c
4   (.25, .5, .75))
5 lm_spline <- lm(AdjSalePrice ~ bs(SqFtTotLiving,
  knots = knots, degree = 3) +
  SqFtTotLot + Bathrooms + Bedrooms + BldgGrade, data =
  house_98105)
```

figure4-12-placeholder.p

Figure 2: Spline Fit

Nonlinear Fit: Splines in Python

- **Practice:** Fits nonlinear price trends in zip 98105
- **Key Insight:** Better matches small vs. large home value shifts

- Curves reflect reality

```
1 # Python (p. 190)
2 import statsmodels.formula.api as smf
3 formula = 'AdjSalePrice~ubs(SqFtTotLiving, udf=6, u
           degree=3)+uSqFtLot+uBathrooms+uBedrooms+u
           BldgGrade'
4 model_spline = smf.ols(formula=formula, data=house_
                        98105).fit()
```

Refining Precision

Model Assessment: Theory

- **Theory (p. 153):** Measures prediction quality and fit
- **RMSE:** $\sqrt{\frac{\sum (y_i - \hat{y}_i)^2}{n}}$ —average error magnitude
- R^2 : Proportion of variance explained (0-1); higher means better fit
- **Use:** Guides housing prediction accuracy

Cross-Validation: Theory

- **Theory (p. 155):** Validates model on unseen data via k -fold splits
- **Process:** Divide data, train on $k - 1$, test on 1, repeat, average RMSE
- **Why:** Ensures predictions generalize beyond training sales—crucial for real estate

Model Selection: Theory & R

- **Theory (p. 156):** Balances fit vs. complexity—Occam's razor
- **AIC:**
 $2P + n \log(\text{RSS}/n)$ —penalizes extra predictors
- **Goal:** Optimal housing model without overkill
- Streamlined predictors

```
1 # R (p. 157)
2 library(MASS)
3 house_full <- lm(AdjSalePrice ~ SqFtTotLiving +
4   SqFtLot + Bathrooms +
5   Bedrooms + BldgGrade + PropertyType, data = house)
6 step <- stepAIC(house_full, direction = "both")
# Drops less impactful vars
```

Model Selection: Python

- **Practice:** Automates predictor choice for housing
- **Key Insight:** Reduces noise, enhances prediction

```
1 # Python (p. 158 adapted)
2 from dmbs import stepwise_selection
3 predictors = ['SqFtTotLiving', 'SqFtTot', '
    Bathrooms', 'Bedrooms', 'BldgGrade']
4 def train_model(vars):
5     model = LinearRegression()
6     model.fit(house[vars], house[outcome])
7     return model
8 best_model, _ = stepwise_selection(house[predictors
    ].columns, train_model)
```

- Focused fit

Weighted Regression: Theory & R

- **Theory (p. 159):** Weights adjust influence by reliability
- **Why:** Older sales less relevant—recent data gets priority
- **Impact:** Refines coefficients for current market
- Recent sales emphasized

```
1 # R (p. 159)
2 house$Weight = year(house$DocumentDate) - 2005
3 house_wt <- lm(AdjSalePrice ~ SqFtTotLiving +
4   SqFtLot + Bathrooms +
5   Bedrooms + BldgGrade, data = house, weight = Weight
6   )
7 # Shifts coeffs slightly
```

Weighted Regression: Python

- **Practice:** Weights tune housing model
- **Key Insight:** Aligns predictions with market trends

- Fresher focus

```
1 # Python (p. 160)
2 house['Weight'] = [int(date.split('-')[0]) for date
3                     in house.DocumentDate] - 2005
4 house_wt = LinearRegression()
5 house_wt.fit(house[predictors], house[outcome],
6              sample_weight=house.Weight)
```

Facing Challenges

Prediction Limits: Theory

- **Theory (p. 161):** Extrapolation beyond data fails—e.g., empty lots
- **Intervals:** Confidence for b_i , wider prediction for \hat{Y}_i
- **Why:** Uncertainty spikes outside training range—limits housing forecasts

Interpreting Coefficients: Theory

- **Theory (p. 171):** Coefficients mislead if predictors correlate
- **Multicollinearity:** Size and bedrooms overlap—unstable fits
- **Confounding:** Missing location skews results
- **Interactions:** Size's effect varies by zip—needs modeling

Diagnostics: Theory & R

- **Theory (p. 176):** Residuals reveal model flaws
- **Outliers:** Extreme sales (e.g., \$119,748); **Influence:** Sway points
- **Heteroskedasticity:** Uneven errors signal gaps

```
1 # R (p. 177)
2 house_98105 <- house[house$ZipCode == 98105, ]
3 lm_98105 <- lm(AdjSalePrice ~ SqFtTotLiving +
4   SqFtLot + Bathrooms +
5   Bedrooms + BldgGrade, data = house_98105)
6 sresid <- rstandard(lm_98105) # -4.326732 outlier
```

figure4-6-placeholder.pd

Figure 3: Influence Plot

Diagnostics: Python

- **Practice:** Identifies \$119,748 as partial sale anomaly
- **Key Insight:** Diagnostics ensure robust housing predictions

```
1 # Python (p. 178)
2 from statsmodels.stats.outliers_influence import
   OLSInfluence
3 house_98105 = house[house['ZipCode'] == 98105]
4 model = smf.ols('AdjSalePrice~_SqFtTotLiving+_
   SqFtLot+_Bathrooms+_Bedrooms+_BldgGrade',
   data=house_98105).fit()
5 influence = OLSInfluence(model)
6 sresiduals = influence.resid_studentized
```

- Spots critical flaws

Insights & Next Steps

- **Findings:** Linear ties price to size, splines capture nonlinear trends
- **Diagnostics:** Reveal quirks like partial sales—critical for accuracy
- **Application:** Real-world tool for buyers, sellers, and assessors

Key Takeaways

- **Flexible:** Evolves from simple lines to complex curves for housing
- **Precise:** RMSE and cross-validation ensure reliable price predictions
- **Powerful:** R and Python implementations unlock data-driven insights

Looking Ahead

- **Resources:** *Statistical Learning* (Hastie et al.), *Time Series Forecasting* (Shmueli)
- **Next Steps:** Dive into splines, time series for dynamic housing models
- **Call:** Blend theory and practice for smarter predictions