Econ 104 Project 1

Cole Kazu Yanagisawa

2025-10-20

0) Setup

To setup the project on the backend, we will use code chunks to load packages, import data, and set up helper functions. Full code chunks can be found in the appendix (see Appendix 0).

---

1) Model Question

This project examines how physical characteristics of a home, such as size, number of bedrooms and bathrooms, and lot area, influence its listed price in the U.S. housing market.
The goal is to understand which property features have the strongest effects on value and whether these effects diminish as homes get larger.

This question matters economically because housing prices affect affordability, property taxes, and household wealth, and they guide investment and development decisions for real estate professionals.
It also provides insight into how buyers and sellers value different home attributes across markets.

Based on economic reasoning, the expected relationships are:

- Larger homes and more bathrooms should increase price, reflecting higher utility and comfort.

- Bigger lots are expected to add value but with diminishing returns
- Bedroom count may have a weaker or even negative relationship when controlling for total house size, since it can trade off with room size or layout quality.

---

2) Descriptive Analysis

2.1 Dataset Citation

Sakib, A. S. (2024). *USA Real Estate Dataset* [Data set]. Kaggle.
https://doi.org/10.34740/KAGGLE/DS/3202774

2.2 Dataset Summary

The dataset originates from Realtor.com and was compiled on Kaggle (2024). Each row represents a single property listing in the United States or its territories, making the unit one home or residential property. The dataset covers 54 states and territories with a total of 44,526 listings (see Appendix 2.2).

Key variables include:

- `price` — listing price
- `bed` — number of bedrooms
- `bath` — number of bathrooms
- `house_size` — interior square footage
- `acre_lot` — land area in acres
- `status` — listing type (e.g., for sale)
- `state` and `city` — property location

Missing data are present in several variables, primarily those describing property features. The share of missing observations varies by variable:

- `bed` (22%), `bath` (23%), `house_size` (26%), `acre_lot` (14%), and `prev_sold_date` (33%)
- `price` (7%), `street` (0.5%), and `brokered_by` (0.2%) show minimal missingness
- `state` and `status` are fully complete

This reflects listings with incomplete property details (such as apartments without lot sizes or missing sale-history data). Because missing values appear random and affect different variables independently, they were retained in the dataset. No imputation, row deletion, or "cleaning" was performed.

2.3 Descriptive Analysis of Variables

2.3.1 Numeric Variables

The numeric variables in this dataset are 'price', 'bed', 'bath', 'house_size', and 'acre_lot.' This section summarizes their central tendencies, dispersion, and relationships through summary statistics and visualizations.

Summary Statistics (see Appendix 2.3)

Table 1: Summary Statistics for Numeric Variables

variable	n	mean	median	sd	min	q1	q3	max
acre_lot	38168	12.81	0.26	638.21	0	0.16	0.98	100000
bath	34135	2.51	2.00	1.52	1	2.00	3.00	99
bed	34745	3.29	3.00	1.60	1	3.00	4.00	102
house_size	33040	2050.15	1760.00	1920.90	140	1300.00	2410.00	217364
price	44495	546523.01	325000.00	4891686.22	0	164900.00	550000.00	1000000000

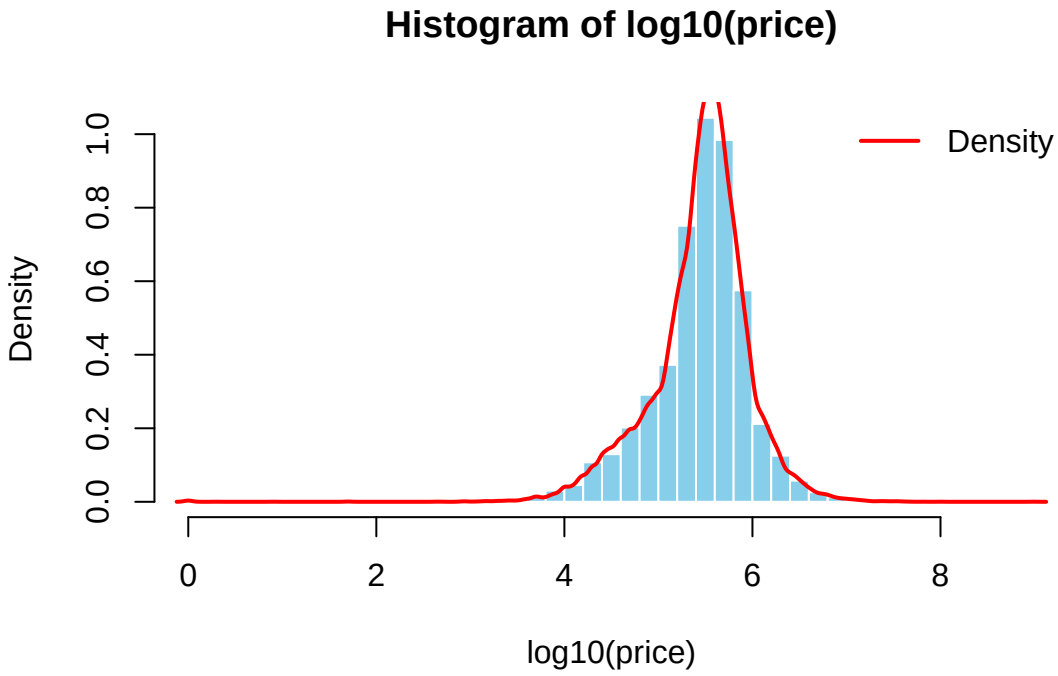
From the summary statistics:

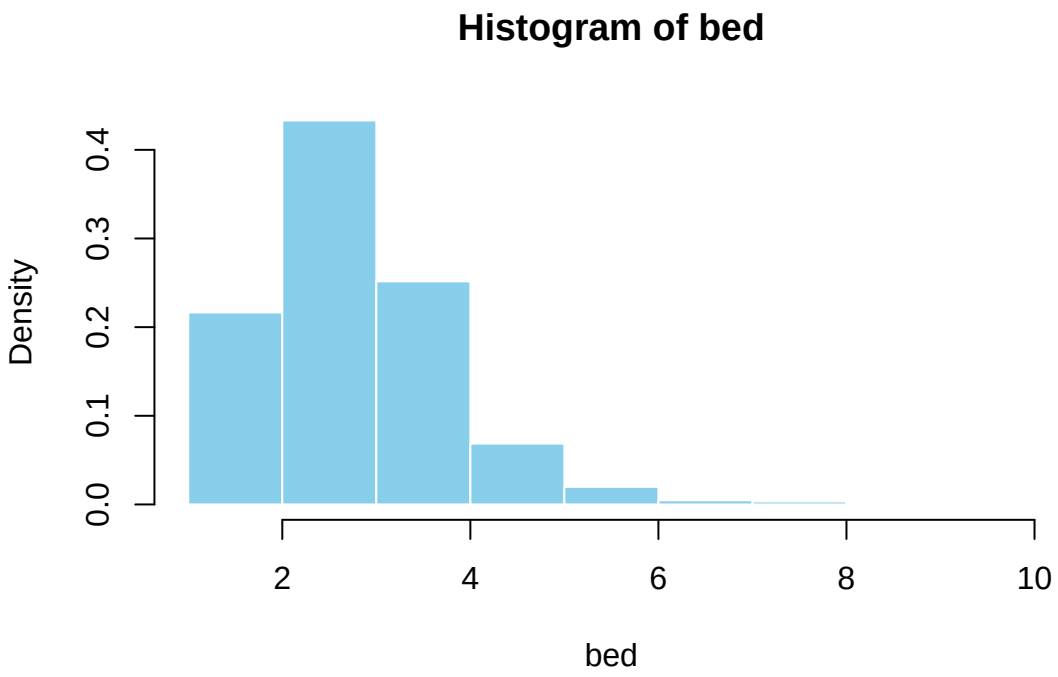
- `Price` shows the greatest range and variation, with values spanning from $0 to $1 billion and a standard deviation of nearly $4.9MM. The mean price ($546,000) is substantially higher than the median ($325,000), indicating a strong right-skew caused by a few extremely high-value listings.
- `House size` also exhibits high variability, ranging from 140 sq ft to over 217 000 sq ft, though most homes are under 2 500 sq ft. The mean (2 050 sq ft) and median (1 760 sq ft) suggest a similar right-skew.
- Lot size (`acre_lot`) is highly skewed, with a mean of 12.8 acres but a median of only 0.26 acres. Most properties occupy less than one acre, but a few very large rural parcels drive the average upward.
- Bedrooms `beds` and bathrooms `baths` have much smaller ranges and standard deviations. Most listings have around three bedrooms and two bathrooms, indicating relatively little variation across homes.

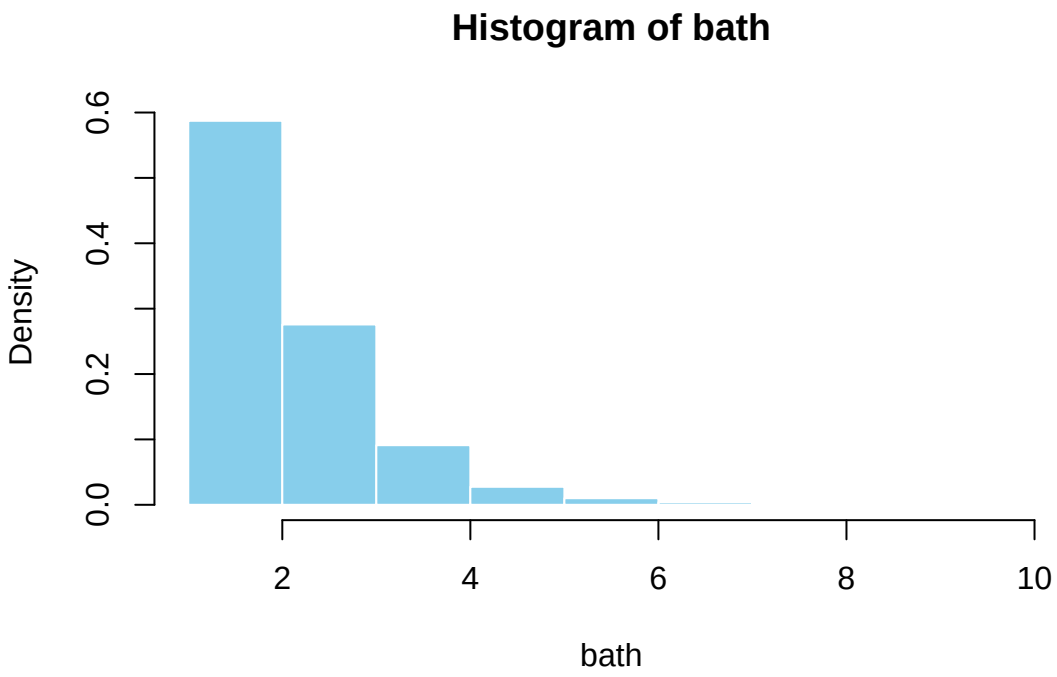
These descriptive measures show that `price`, `house_size`, and `acre_lot` are strongly right-skewed and contain several large outliers, while `bed` and `bath` are more tightly distributed around their central values. These patterns justify considering log transformations for `price` and `acre_lot` in later modeling to reduce skewness and stabilize variance.

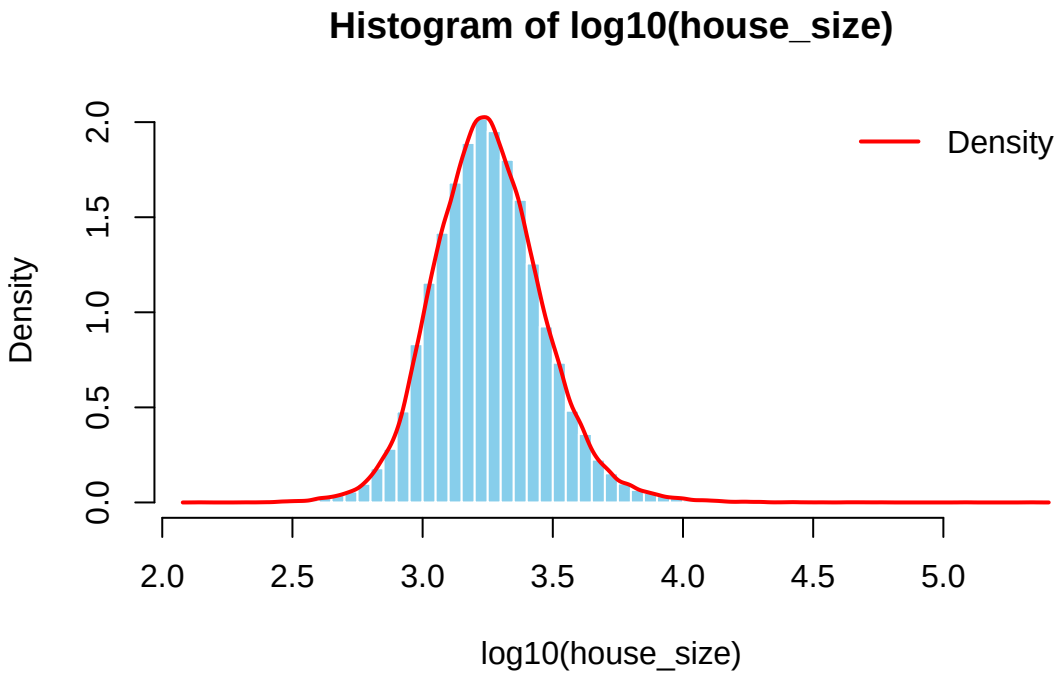
Histograms, Distribution Shapes, and Fitted Disributions (see Appendix 2.4)

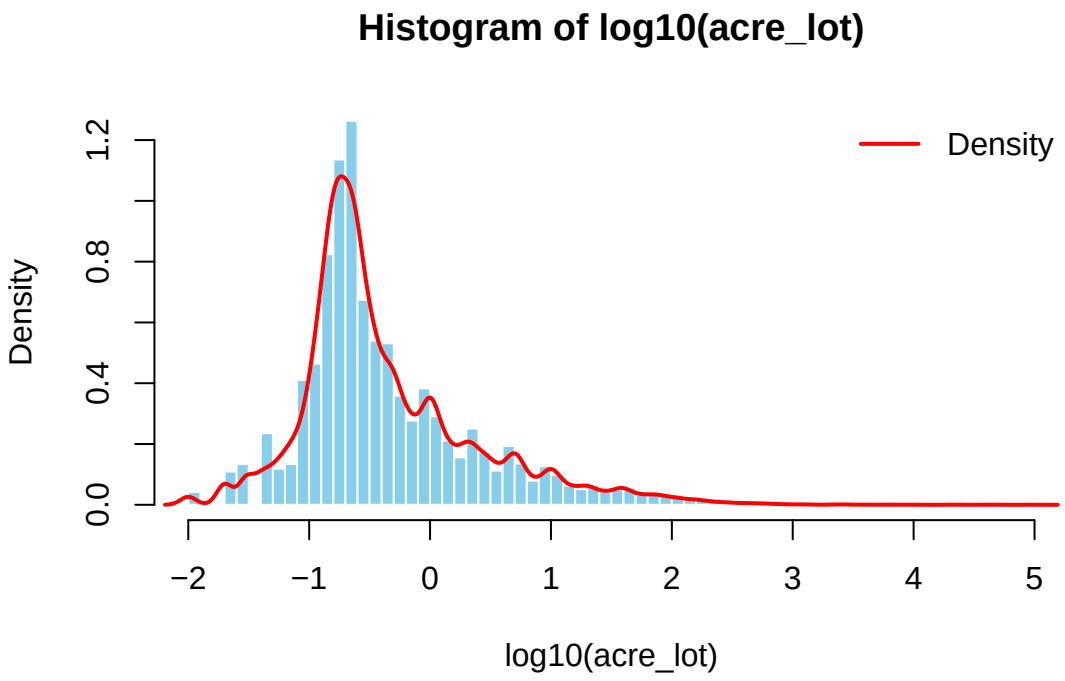
The following histograms illustrate the distributions of the five numeric variables in the dataset: `price`, `bed`, `bath`, `house_size`, and `acre_lot`.
These plots help visualize the overall shape of each variable and reveal any skewness or outliers that were also suggested by the summary statistics.


Histogram of log10(price)
Density
0.0
0.2
0.4
0.6
0.8
1.0
0
2
4
6
8
log10(price)
Density


Histogram of bed
Density
0.0
0.1
0.2
0.3
0.4
2
4
6
8
10
bed


Histogram of bath
Density
0.0
0.2
0.4
0.6
2
4
6
8
10
bath


Histogram of log10(house_size)
Density
Density
0.0
0.5
1.0
1.5
2.0
2.0
2.5
3.0
3.5
4.0
4.5
5.0
log10(house_size)



For discrete variables (`bed` and `bath`), density lines not applicable; observations exceeding 10 were truncated for plotting purposes ($n = 91$ for bedrooms; $n = 58$ for bathrooms). These outliers remain in the dataset for analysis.

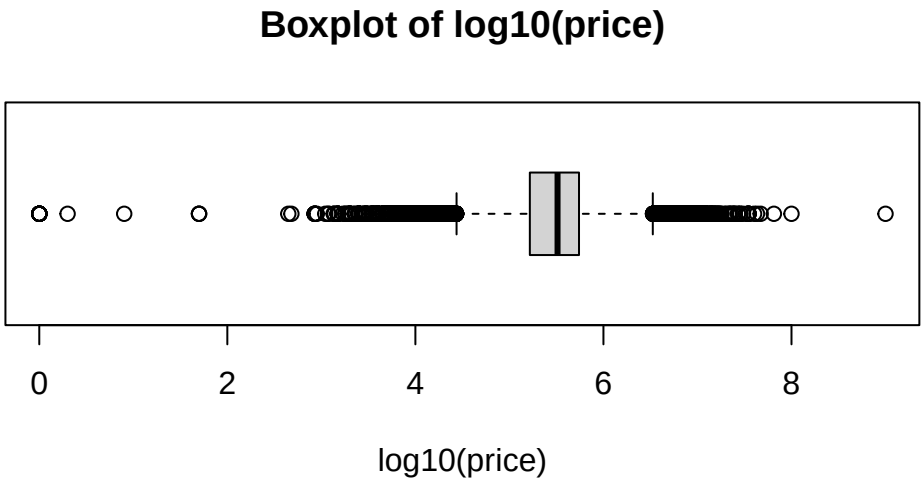
The histograms show that several numeric variables in the dataset are highly skewed, while others are relatively symmetric.

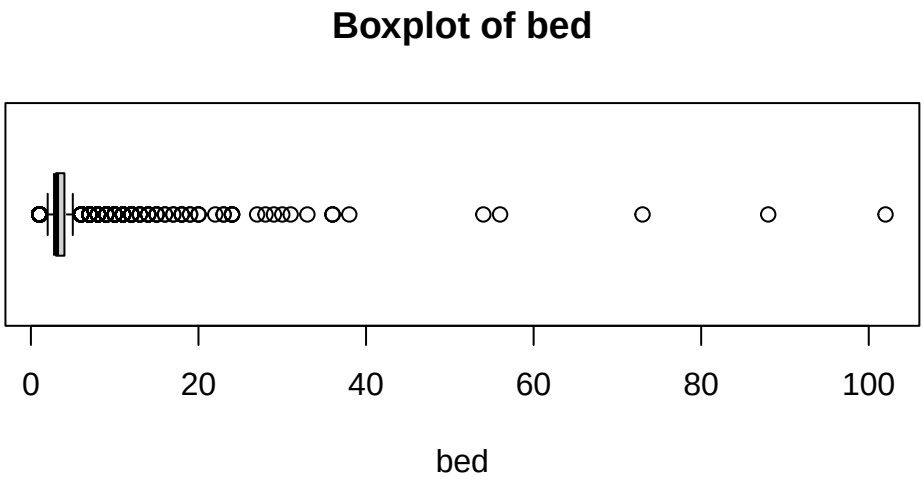
- `price` and `house_size` both become approximately bell-shaped when plotted on a log scale, confirming that their distributions are right-skewed on the raw scale due to a small number of very high-priced or large properties.
- Lot size `acre_lot` remains somewhat right-skewed even after log transformation, suggesting that some extremely large rural parcels influence the distribution.
- **Bedrooms `beds` and bathrooms `bath` are discrete variables with small integer values, centered around three bedrooms and two bathrooms. Their shapes are roughly symmetric and reflect typical residential housing patterns. There are values that extend well beyond the typical range for a single family residence, which likely represent multi-unit properties or commercial listings.

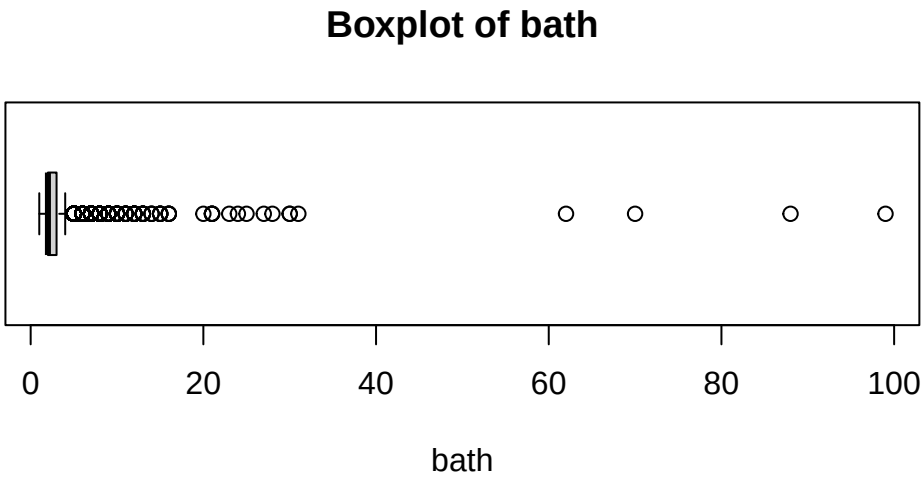
The log transformation improves the visibility and interpretability of `price`, `house_size`, and `acre_lot`, showing that most listings cluster around moderate values while outliers stretch the upper tail.

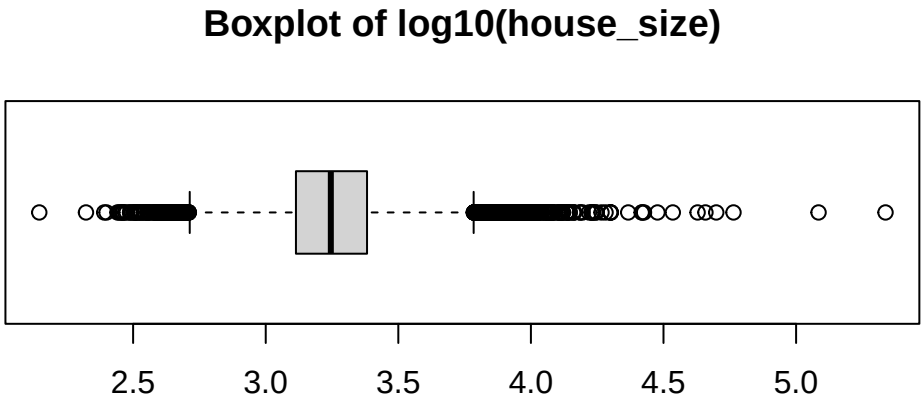
Boxplots Boxplots visualize the five-number summary (minimum, Q1, median, Q3, maximum) and highlight outliers.

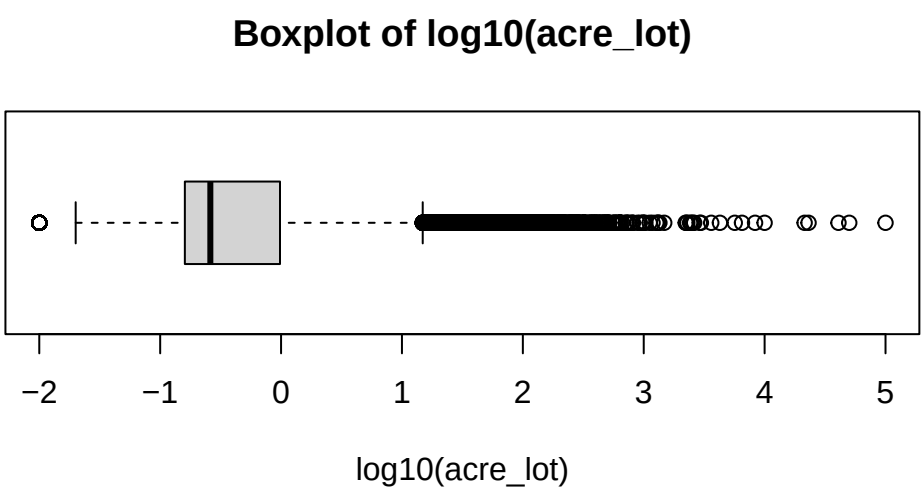
Because some variables have extreme right tails, the plots automatically switch to a log scale for readability (for visualization only) (see Appendix 2.5).


Boxplot of log10(price)
0
2
4
6
8
log10(price)


Boxplot of bed
0
20
40
60
80
100
bed


Boxplot of bath
0
20
40
60
80
100
bath


Boxplot of log10(house_size)
2.5
3.0
3.5
4.0
4.5
5.0
log10(house_size)



The boxplots reaffirm the high variability and presence of outliers among several numeric variables (seen in histograms).

On the log scale, both `price` and `house_size` are roughly symmetric, telling us that the log transformation effectively stabilizes their variance.

`acre_lot` is still right-skewed, showing that some listings have disproportionately large parcels.

`bed` and `bath` have very narrow interquartile ranges centered on typical residential setups (around 3 bedrooms and 2 bathrooms), though a few outliers likely reflect data entry errors or very large multifamily properties.

These plots reinforce that log transformations are suitable for `price` and `house_size`, while `bed` and `bath` can be treated as discrete count variables.

Correlation Matrix The correlation matrix tells us how the numeric variables relate to one another.

It helps identify which variables are most strongly associated with `price`, and whether any variables are highly correlated with each other, which could indicate multicollinearity later (see Appendix 2.6).

Table 2: Correlation Matrix of Numeric Variables

	price	bed	bath	house_size	acre_lot
price	1.00	0.26	0.40	0.32	0.01
bed	0.26	1.00	0.78	0.49	0.00
bath	0.40	0.78	1.00	0.55	0.00
house_size	0.32	0.49	0.55	1.00	0.00
acre_lot	0.01	0.00	0.00	0.00	1.00

The correlation matrix shows that several variables move together in predictable ways.
`price` is most closely related to `house_size` (r 0.32), meaning larger homes tend to sell for more, though the relationship isn't extremely strong.
`bed` and `bath` are also linked to `price`, but mainly because they're both tied to the overall size of the house. The strong correlation between `bed` and `bath` (r 0.78) makes sense because houses with more bedrooms usually have more bathrooms too.
`acre_lot` stands out for having almost no relationship with the other variables, suggesting that land size doesn't have much influence on listing price in this sample.

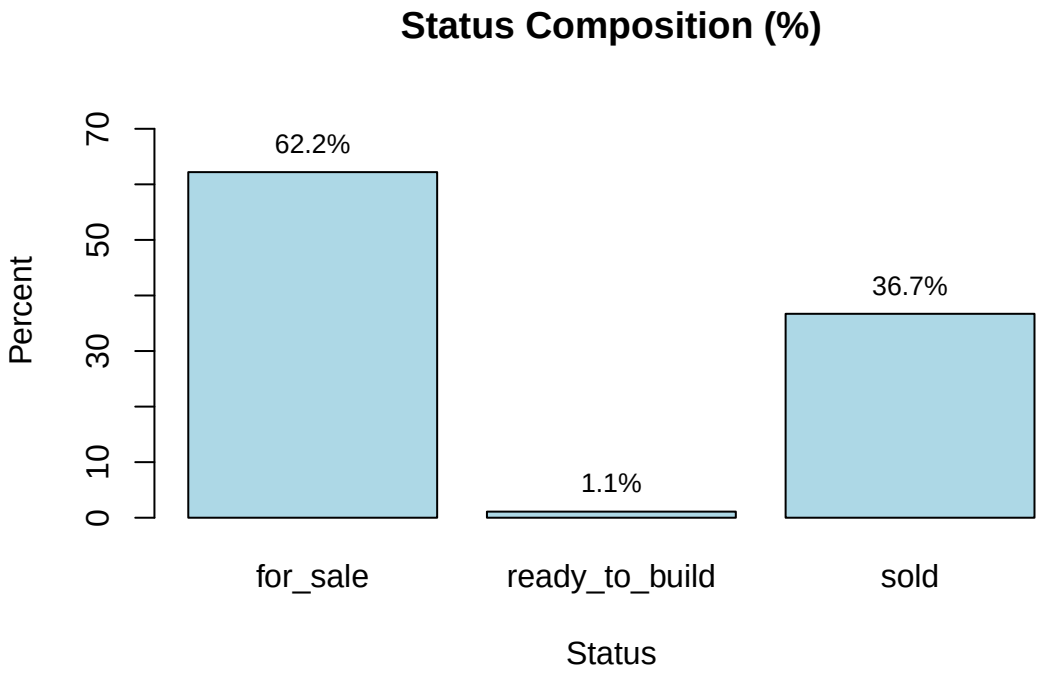
After looking at the matrix, it looks like interior space and room count are the biggest drivers of price differences, while lot size plays a smaller role.

Summary of Findings This section has described the numeric variables in detail. Key takeaways include:

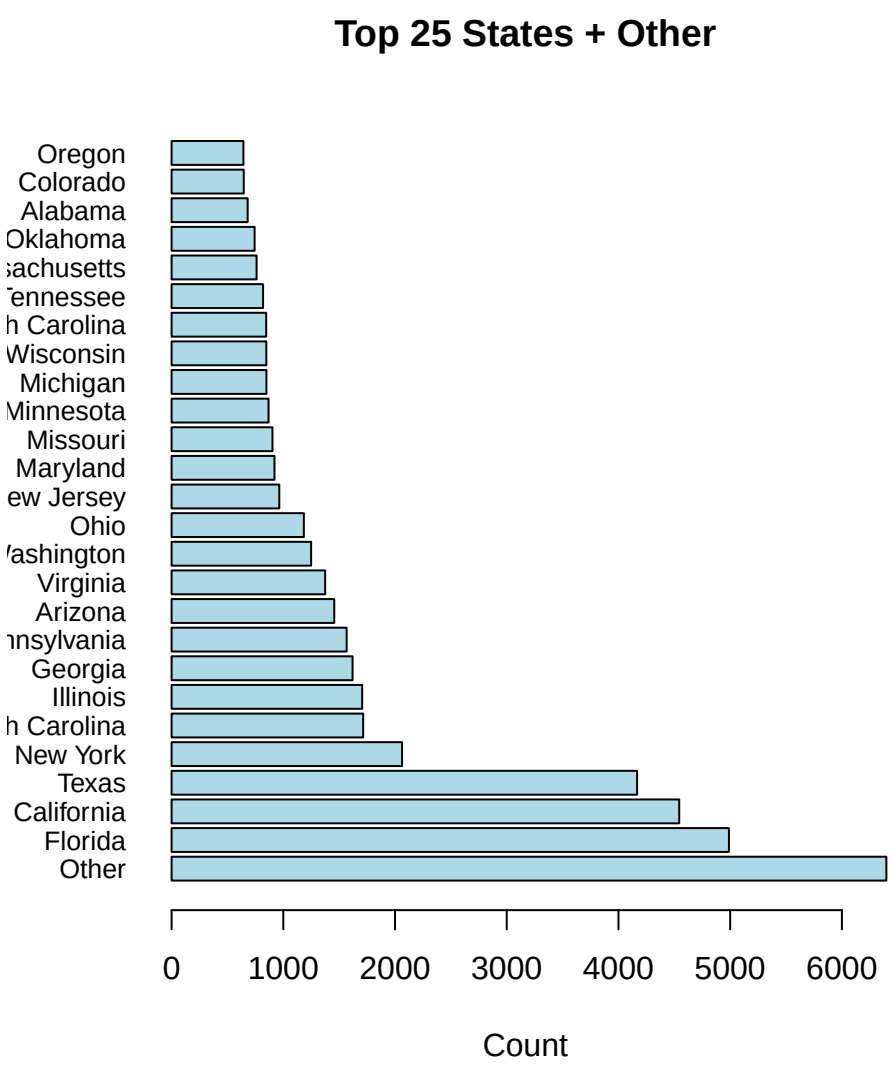
- `price`, `house_size`, and `acre_lot` are strongly right-skewed, justifying log transformations for modeling.
- `bed` and `bath` are more tightly clustered around typical values, reflecting common residential configurations.
- `price` is most strongly correlated with `house_size`, while `bed` and `bath` are highly correlated with each other, indicating potential multicollinearity concerns.
- `acre_lot` shows little correlation with other variables, suggesting it may not be a key driver of price in this dataset. These insights will guide model specification and variable transformations in later sections.

2.3.2 Categorical Variables

Categorical variables summarize the distribution of listings across different groups. (see Appendix 2.7)



State Composition (see Appendix 2.8)



Most listings are concentrated in a few states that dominate the dataset (California, Florida, Texas). Grouping the remaining smaller states into "Other" keeps the figure readable while still showing the overall spread of listings.

2.4 Regression Assumption Hints / Possible Violations

The descriptive analysis gives a few clues about what to watch for in the regression stage:

- `price` and `acre_lot` are highly right-skewed. A log transformation will likely help normalize them.
- `house_size`, `bed`, and `bath` are all moderately to strongly correlated, which could lead to multicollinearity.

- Several extreme outliers exist in `price` and `house_size`, which might pull the regression line toward them.
- Because larger homes tend to have both higher prices and more variation, heteroskedasticity (non-constant error variance) is likely.
- If we include categorical variables such as `state` or `status`, fixed effects might capture much of the between-region variation, improving model fit.

These insights will inform model specification, variable transformations, and diagnostic tests in the following sections.

2.5 Train/Test Split

Before running any regression, the dataset was split into training and testing subsets.
The training set (80%) will be used to fit the model, while the testing set (20%) will be used to evaluate how well the model generalizes to unseen data.
Random sampling ensures each subset represents the overall dataset fairly.

Now we need to run a few tests to make sure our split worked well and we use it for our model (see Appendices 2.9 and 2.10).

Note: During setup, a random seed was set using `set.seed(104)` to make the split reproducible.

```
# 80/20 train-test split
train_indices <- sample(1:nrow(df), size = 0.8 * nrow(df))
train_set <- df[train_indices, ]
test_set  <- df[-train_indices, ]

# Check sizes
nrow(train_set); nrow(test_set)
```

```
## [1] 35620
```

```
## [1] 8906
```

```
# Compare key variable (price)
summary(train_set$price)
```

```
##       Min.    1st Qu.     Median       Mean    3rd Qu.       Max.       NA's
##          0     163408     325000     552873     550000 1000000000         25
```

```
summary(test_set$price)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.     NA's
##        8   165000   325000   521125   550000 34995000        6
```

```r
# Quick numeric check
train_mean <- mean(train_set$price, na.rm = TRUE)
test_mean  <- mean(test_set$price, na.rm = TRUE)
train_median <- median(train_set$price, na.rm = TRUE)
test_median  <- median(test_set$price, na.rm = TRUE)

cat("Train mean:", round(train_mean, 0),
    " | Test mean:", round(test_mean, 0), "\n")
```

```
## Train mean: 552873  | Test mean: 521125
```

```r
cat("Train median:", round(train_median, 0),
    " | Test median:", round(test_median, 0))
```

```
## Train median: 325000  | Test median: 325000
```

```r
# Compare percent of listings by state in each split
round(100 * prop.table(table(train_set$state)), 1)[1:5]
```

```
##
##    Alabama     Alaska    Arizona   Arkansas California
##        1.5        0.1        3.2        1.1       10.2
```

```r
round(100 * prop.table(table(test_set$state)), 1)[1:5]
```

```
##
##    Alabama     Alaska    Arizona   Arkansas California
##        1.7        0.1        3.6        0.9       10.0
```

Conclusions from the Tests: The 80/20 split worked well. The training set has about 35,600 listings, and the test set has around 8,900.
Both sets show nearly identical price distributions, with medians around $325,000 and similar means.
That tells us the random split kept things balanced and that the training and test data represent the overall dataset pretty evenly.
The proportions by state are also consistent, so no single region dominates one subset more than the other.
The split looks reasonable and should give reliable results when we test the model later.

---

3) Baseline Linear Regression Model

Purpose: This section fits a simple multiple linear regression model to estimate how home features relate to price.
The goal is to see which variables most influence price.

3.1 Model Form and Rationale

The baseline model includes the main structural variables identified in the descriptive analysis.
Because `price` is highly right-skewed, the dependent variable is transformed using the natural logarithm to make the distribution more normal and to interpret effects as approximate percentage changes.

Model:

$$\log(\text{price}_i) = \beta_0 + \beta_1 \text{bed}_i + \beta_2 \text{bath}_i + \beta_3 \text{house\_size}_i + \beta_4 \text{acre\_lot}_i + \epsilon_i$$

3.2 Estimation

(see Appendix 3.2)

```
##
## Call:
## lm(formula = log(price) ~ bed + bath + house_size + acre_lot,
##     data = train_set)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -15.6328  -0.3853  -0.0075   0.3921   3.8315
##
## Coefficients:
##                Estimate  Std. Error t value            Pr(>|t|)
## (Intercept) 12.15374010  0.01291723 940.893 <0.0000000000000002 ***
## bed         -0.07794685  0.00526236 -14.812 <0.0000000000000002 ***
## bath         0.34480517  0.00546582  63.084 <0.0000000000000002 ***
## house_size   0.00003968  0.00000298  13.315 <0.0000000000000002 ***
## acre_lot     0.00002409  0.00001194   2.018              0.0436 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7227 on 21867 degrees of freedom
##   (13748 observations deleted due to missingness)
## Multiple R-squared:  0.286,  Adjusted R-squared:  0.2859
## F-statistic:  2190 on 4 and 21867 DF,  p-value: < 0.00000000000000022
```

Interpretation: The baseline regression models the natural log of price using key structural features: bedrooms, bathrooms, house size, and lot size.
All four variables are statistically significant, and most signs make sense given what we saw earlier in the descriptive analysis.

- **Intercept (12.15):** Represents the expected log(price) when all predictors are zero. This is the baseline level.
- **Bedrooms (–0.078):** Holding other factors constant, homes with more bedrooms tend to have slightly lower prices. Maybe adding bedrooms reduces average room size or reflects smaller layouts.

- **Bathrooms (+0.345):** Each additional bathroom is associated with roughly a 35% higher price (`exp(0.345)  1.41`), ceteris paribus.

- **House size (+0.0000397):** A one-square-foot increase raises price by about 0.004%. Interpreted more practically, every extra 1,000 sq. ft. adds about 4% to price.

- **Lot size (+0.000024):** The effect is positive but very small — larger lots contribute slightly to value, though not as much as interior space.

Our basic model, using only only structural characteristics, explains about 29% of the variation in log(price) (R^2 = 0.286). The residual standard error (~0.72) signals a moderate spread around predicted values, and the F-statistic confirms the model is highly significant overall (p < .001).
Our model supports what we saw earlier–interior features, especially bathrooms and total square footage, drive most of the differences in housing prices.

---

4) Multicollinearity Check (VIF)

Variance Inflation Factors (VIF) measure how strongly each variable is correlated with the others in the model.
High VIF values (> 5) signal multicollinearity, so two or more (variables explain similar variation in price) which can make coefficient estimates unreliable.

4.1 VIF Calculation

(see Appendix 4.1)

```
##              Variable  VIF
## bed               bed 2.23
## bath             bath 2.47
## house_size house_size 1.45
## acre_lot     acre_lot 1.00
```

All predictors show acceptable VIF values (< 5), confirming no major multicollinearity. The overlap between bedrooms and bathrooms is expected but not high enough to distort the model.

---

5) Model Refinement (AIC / BIC Selection)

This step refines the baseline regression model using an information-criteria approach to balance fit and simplicity.
Lower AIC (Akaike Information Criterion) or BIC (Bayesian Information Criterion) values indicate a better trade-off between explanatory power and the number of predictors.

5.1 Stepwise Selection

The model was refined using a stepwise procedure based on AIC, which tests different combinations of predictors and selects the one with the lowest AIC value (see Appendix 5.1 and 5.2).

```
# Stepwise selection by AIC
model_refined <- MASS::stepAIC(model1, direction = "both", trace = FALSE)
summary(model_refined)
```

```
##
## Call:
## lm(formula = log(price) ~ bed + bath + house_size + acre_lot,
##     data = train_set)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -15.6328  -0.3853  -0.0075   0.3921   3.8315
##
## Coefficients:
##                Estimate  Std. Error t value            Pr(>|t|)
## (Intercept) 12.15374010  0.01291723 940.893 <0.0000000000000002 ***
## bed         -0.07794685  0.00526236 -14.812 <0.0000000000000002 ***
## bath         0.34480517  0.00546582  63.084 <0.0000000000000002 ***
## house_size   0.00003968  0.00000298  13.315 <0.0000000000000002 ***
## acre_lot     0.00002409  0.00001194   2.018              0.0436 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7227 on 21867 degrees of freedom
##   (13748 observations deleted due to missingness)
## Multiple R-squared:  0.286,  Adjusted R-squared:  0.2859
## F-statistic:  2190 on 4 and 21867 DF,  p-value: < 0.00000000000000022

## Baseline AIC: 47869.03

## Refined AIC: 47869.03

## Baseline BIC: 47916.99

## Refined BIC: 47916.99
```

5.2 Results and Comparison

The AIC and BIC values for the refined model are identical to those of the baseline model, meaning no predictors were added or removed during the stepwise process. In other words, the algorithm found that dropping or adding any variable would worsen the model's information criteria.

Summary of Findings:

- Method used: Stepwise selection using AIC via MASS::stepAIC().
- Variables retained: bed, bath, house_size, and acre_lot.
- AIC/BIC: Unchanged

- R^2 and Adjusted R^2: Essentially unchanged (0.286).
- Significance patterns: All variables remained statistically significant with the same coefficient signs as before.

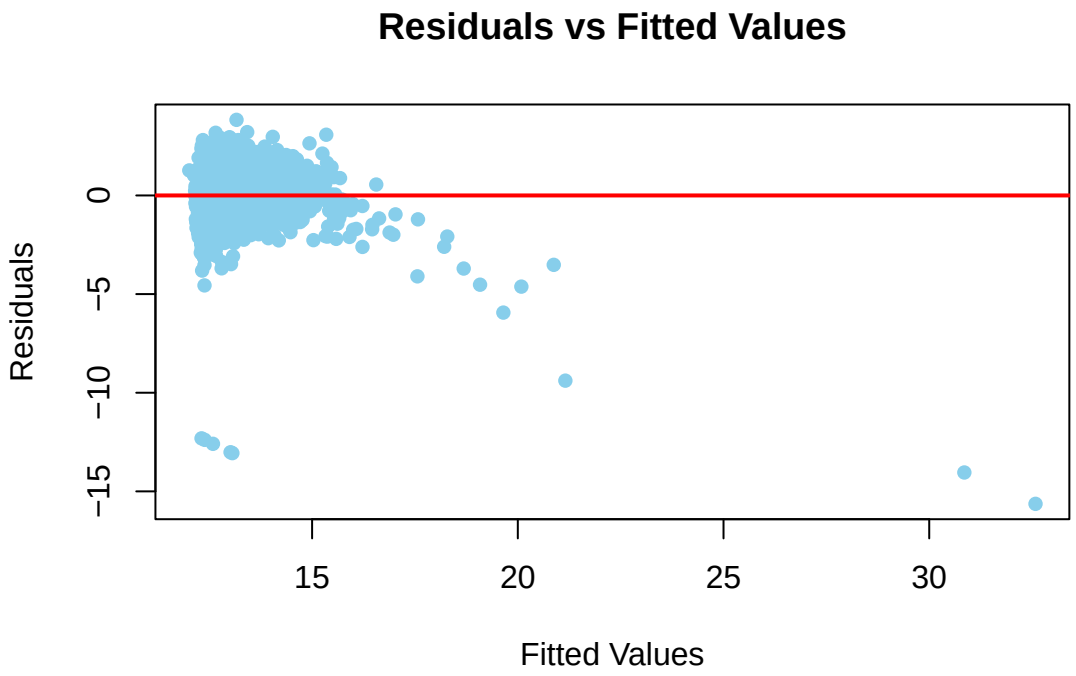
---

6) Residual Analysis

We need to chek if the regression assumptions hold by analyzing the residuals. This ensures our model is valid and reliable.

6.1 Residual vs Fitted Plot

This plot checks whether residuals are randomly scattered around zero, which would indicate that the model fits evenly across the range of predicted values (see Appendix 6.1).



Interpretation:

Residuals appear roughly centered around zero for most fitted values, indicating that the model does not show major bias across typical price levels; however, as fitted values increase, residuals trend slightly downward and become more dispersed, suggesting slight heteroskedasticity. The model fits low-priced and mid-priced homes better than high-priced ones.
A few large negative residuals also indicate potential outliers.

---

7) RESET Test

The Ramsey RESET test checks whether the model is missing nonlinear terms or interactions. Per the rubric, we test quadratic terms and two-way interactions by setting `power = 2` on the regressors.
We also report the broader default test (`power = 2:3`) for reference.

7.1 RESET (quadratic)

```
##
##  RESET test
##
## data:  model1
## RESET = 636.55, df1 = 4, df2 = 21863, p-value < 0.00000000000000022
```

This provides strong evidence *against* the null hypothesis of correct functional form.

Interpretation:
The model likely omits nonlinear relationships or interaction effects.
In a housing context, this could mean variables like `house_size` or `bath` have diminishing returns (nonlinear impact), or that combinations such as `bed × bath` jointly affect price.
A natural next step would be to explore adding quadratic terms (e.g., `I(house_size^2)`) or log-transformed interactions to better capture these patterns.

7.2 RESET (default powers 2 and 3)

The default RESET test extends the previous one by also including cubic terms (`power = 2:3`).
This version checks for both quadratic and cubic nonlinearities, as well as two- and three-way interactions among regressors.

```
##
##  RESET test
##
## data:  model1
## RESET = 407.6, df1 = 8, df2 = 21859, p-value < 0.00000000000000022
```

The test again rejects the null hypothesis of correct functional form.

Interpretation:
Even when allowing for cubic terms, the model still shows signs of misspecification.
This suggests that additional nonlinear transformations or interaction effects may be needed. For example, squared terms like `I(house_size^2)` or interactions such as `bath × house_size`.
Given the context of housing data, this is not unexpected since prices often rise with house size and amenities but at a decreasing rate.

7.3 Model Adjustment

To address the results of the RESET tests, quadratic terms were added for the key continuous variables (`house_size` and `acre_lot`).
These capture diminishing returns for property size and lot area.

```
##
## Call:
## lm(formula = log(price) ~ bed + bath + house_size + I(house_size^2) +
##     acre_lot + I(acre_lot^2), data = train_set)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -13.2273  -0.3680  -0.0101   0.3766   3.2838
##
## Coefficients:
##                         Estimate       Std. Error t value             Pr(>|t|)
## (Intercept)     12.1479407005822  0.0125835644697 965.382 < 0.0000000000000002
## bed             -0.1106716370417  0.0052183510873 -21.208 < 0.0000000000000002
## bath             0.2282228666248  0.0063245591691  36.085 < 0.0000000000000002
## house_size       0.0002396982183  0.0000065575786  36.553 < 0.0000000000000002
## I(house_size^2) -0.0000000011436  0.0000000000336 -34.032 < 0.0000000000000002
## acre_lot         0.0001773553718  0.0000353479608   5.017          0.000000528
## I(acre_lot^2)   -0.0000000038328  0.0000000008197  -4.676          0.000002946
##
## (Intercept)     ***
## bed             ***
## bath            ***
## house_size      ***
## I(house_size^2) ***
## acre_lot        ***
## I(acre_lot^2)   ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7039 on 21865 degrees of freedom
##   (13748 observations deleted due to missingness)
## Multiple R-squared:  0.3227, Adjusted R-squared:  0.3225
## F-statistic:  1736 on 6 and 21865 DF,  p-value: < 0.00000000000000022

## Baseline AIC: 47869.03

## Quadratic model AIC: 46719.73
```

Interpretation:

Adding the squared terms improved overall model fit. AIC dropped, and adjusted R^2 increased (0.29 to 0.32). The negative coefficients on the squared terms confirm diminishing marginal returns. Larger homes and lots still increase price, but at a decreasing rate.

7.4 RESET Re-Test After Correction

To verify that the quadratic terms addressed the functional form issues, the RESET test was rerun on the updated model.

```
##
##  RESET test
##
## data:  model2
## RESET = 258.76, df1 = 6, df2 = 21859, p-value < 0.00000000000000022
```

The RESET statistic fell (from 636 to 259) though the p-value remains < 0.001, meaning some nonlinearities or interactions may still exist.

Interpretation:

The adjustment improved the model overall, but a some nonlinearity remains. Further analysis might include adding interaction terms.

---

8) Heteroskedasticity Tests and Feasible Corrections

Residual plots (Section 6) showed a slight funnel pattern , with variance increasing with fitted values. This suggests possible heteroskedasticity.
Because this pattern affects the entire model rather than one specific indicator variable, the Breusch–Pagan test is appropriate.

8.1 Breusch–Pagan (BP) Test

The BP test checks whether the variance of residuals depends on the regressors.

```
##
##  studentized Breusch-Pagan test
##
## data:  model2
## BP = 1260.6, df = 6, p-value < 0.00000000000000022
```

8.2 Goldfeld-Quandt (GQ) Test

The GQ test checks for heteroskedasticity by splitting the data based on a chosen variable (here, `house_size`) and comparing variances.

```
##
##  Goldfeld-Quandt test
##
## data:  model2
## GQ = 0.82725, df1 = 10929, df2 = 10929, p-value = 1
## alternative hypothesis: variance increases from segment 1 to 2
```

Interpretation:
The GQ test finds no evidence that heteroskedasticity increases specifically with house size, nmeaning variance in residuals is not explained by this single variable.
Combined with the significant Breusch–Pagan result, this implies heteroskedasticity likely depends on several regressors rather than any one feature.

8.3 Correction Using Robust Standard Errors

Given the evidence of heteroskedasticity from the BP test, the model was re-estimated with White's heteroskedasticity-robust (HC1) standard errors.

```
##
## t test of coefficients:
##
##                          Estimate        Std. Error  t value
## (Intercept)     12.14794070058224  0.08886761000913 136.6971
## bed             -0.11067163704175  0.02139635705650  -5.1725
## bath             0.22822286662479  0.02841725618355   8.0311
## house_size       0.00023969821831  0.00002429139142   9.8676
## I(house_size^2) -0.00000000114362  0.00000000012347  -9.2625
## acre_lot         0.00017735537177  0.00005791608763   3.0623
## I(acre_lot^2)   -0.00000000383284  0.00000000123397  -3.1061
##                             Pr(>|t|)
## (Intercept)     < 0.00000000000000022 ***
## bed              0.000000233078967395 ***
## bath             0.000000000000001014 ***
## house_size      < 0.00000000000000022 ***
## I(house_size^2) < 0.00000000000000022 ***
## acre_lot                    0.002199 **
## I(acre_lot^2)               0.001898 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Interpretations:

After adjusting the model using White's robust standard errors, all variables stayed significant and pointed in the same direction as before. That means the uneven variance we saw earlier affected the precision of the estimates a bit, but not the overall story.

The squared terms for house_size and acre_lot are still negative and significant, which fits the idea of diminishing returns. (As properties get bigger, each extra square foot or acre adds less to the price.) The small negative coefficient on bed probably comes from its overlap with bath and house_size rather than meaning bedrooms actually lower price.

The robust model shows that the main patterns hold up well even after accounting for heteroskedasticity.

8.4 Summary of Findings

Both the Breusch–Pagan and Goldfeld-Quandt tests were used to check for heteroskedasticity. The BP test indicated heteroskedasticity is present, while the GQ test found no evidence that it depends solely on house size. To address this, the model was re-estimated using White's heteroskedasticity-robust standard errors (HC1). The robust model confirmed that all predictors remain significant with similar coefficients, indicating that the main relationships are stable even after accounting for heteroskedasticity.

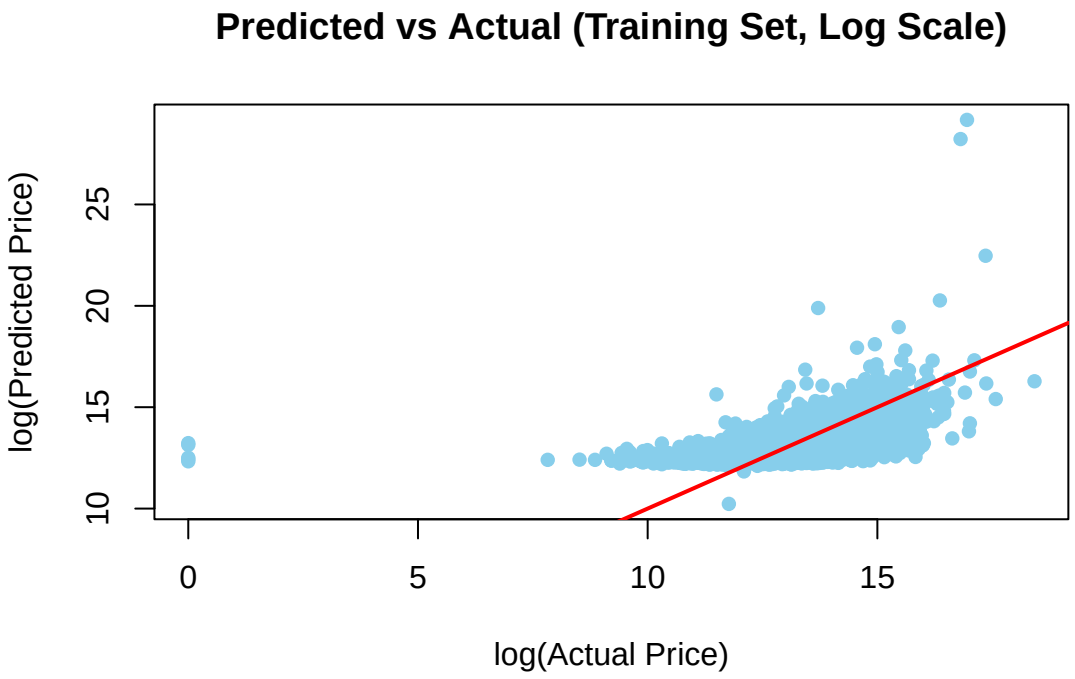
---

9) Final Model

Now that the model has been refined and adjusted for heteroskedasticity, this section reviews how well it performs on the training data and decides which version will serve as the final model for evaluation.

9.1 In-Sample Fit

The adjusted R^2 and AIC from earlier already give a sense of how well the model explains variation in log(price).
A quick look at predicted vs. actual prices helps visualize how closely the fitted values match the data used for estimation.



The plot above shows how well the model predicts home prices (on the log scale) using the training data. Most points cluster near the red 45° line, meaning the model does a decent job predicting average-priced homes.

At the high end, predictions start to spread out, showing that the model tends to under predict luxury properties, which is expected since extreme prices are harder to capture with simple variables

9.2 Model Performance Metrics

The following performance metrics assess the overall fit of Model2 (the quadratic model) on the training set, quantifying the trade-off between its explanatory power (Adj.R^2) and model complexity (AIC/BIC) before moving to final comparison.

```
## Adjusted R²: 0.322
```

```
## AIC: 46719.73
```

```
## BIC: 46783.68
```

After refining the model by adding quadratic terms for house_size and acre_lot, the fit improved noticeably. The adjusted R^2 of 0.32 means the model explains about one-third of the variation in logged housing prices

(slightly better than the baseline version). Both AIC (46,720) and BIC (46,784) dropped compared to the baseline, which tells us this version strikes a better balance between accuracy and simplicity.

Economically, the changes make sense:

- Larger homes and bigger lots still increase price, but at a slower rate. This is captured by the negative squared terms (diminishing returns).
- Bathrooms have the strongest positive effect among all features, which aligns with how amenities drive value.
- The small negative coefficient on bedrooms likely reflects overlap with house size and bathrooms rather than a true price penalty.

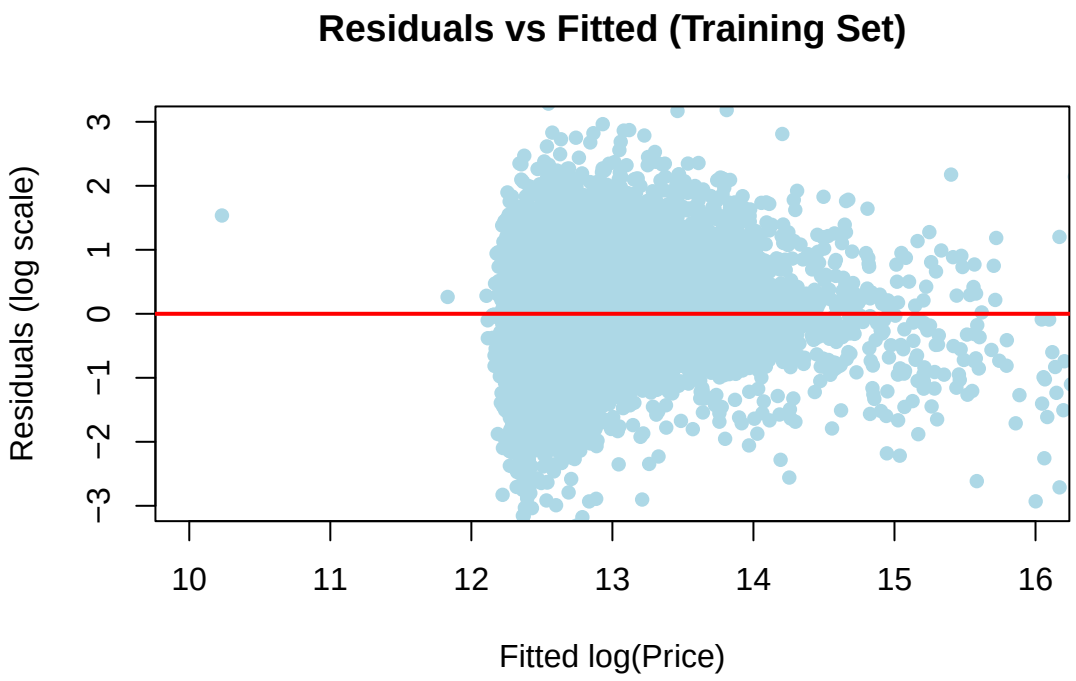
9.2.1 Model comparison: quadratic vs interaction The following comparison assesses whether the inclusion of bath:house_size interaction provides a statistically significant improvement in model fit and complexity compared to the quadratic model. The lowest AIC and BIC values indicate the preferred model.

Table 3: Model comparison: interaction

Model	AIC	BIC	AdjR2
Quadratic only	46719.73	46783.68	0.322
Quadratic + bath:house_size	45443.01	45514.95	0.361

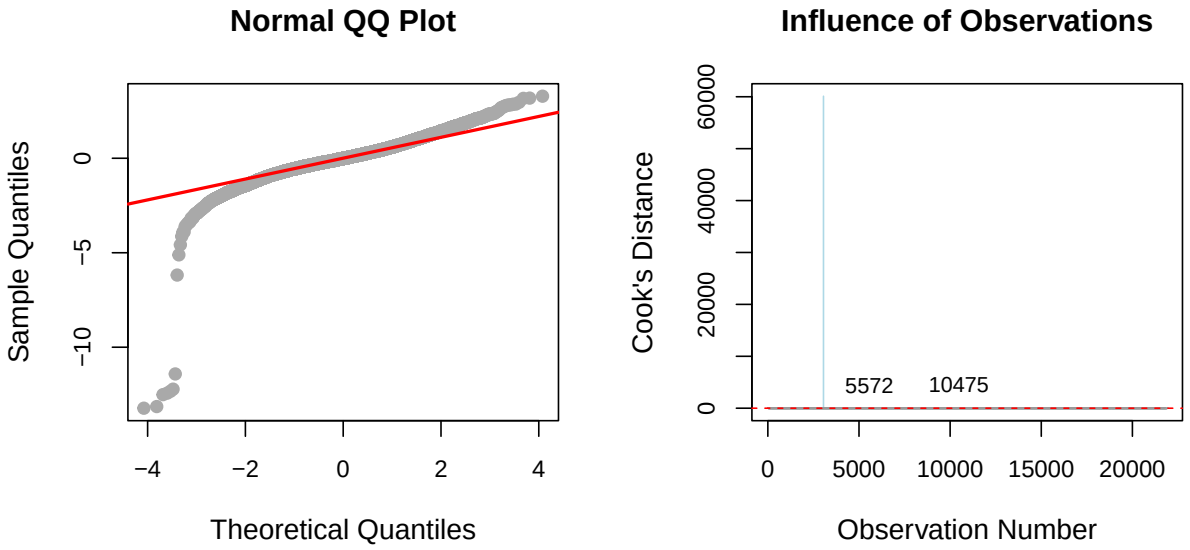
Comparing AIC/BIC and adjusted R^2 indicates that the model including bath:house_size interaction provides a stronger fit,implying that the effect of bathrooms on housing prices depends on the overall size of the home

9.3 Residual Diagnostics



The model does a good job predicting mid-range homes. At the higher end, the errors get wider, meaning the model has a tougher time with very expensive properties which is a common issue in housing data where luxury listings behave differently. There is a bit of a funnel shape where residuals spread out more for higher prices, which suggests heteroskedasticity and is consistent with earlier tests. There's no clear curvature or wave pattern, which means our functional form (including the squared terms) captures the relationship well.

9.3.1 QQ Plot and Cook's Distance Diagnostics The QQ Plot and Cook's Distance check for normality of errors and help to identify highly influential outliers. This is confirmation that the robust standard errors applied in Section 8.3 are necessary to account for the impact of extreme values on the model's stability.



9.3.2 Out-of-sample errors in dollars To fully evaluate the final model and provide economically meaningful error metrics, the out-of-sample predictions are back-transformed from the log scale to dollar terms using the Duan smearing factor. This section calculates the untrimmed, median, and trimmed dollar-term errors from Model2_int's predictions.

```
## mae ($): $49,832,817,175
## rmse ($): $3,654,794,928,461
```

```
## Median AE ($): $172,271
## Trimmed MAE (<=99.5th pct): $50,084,129,183
## Trimmed RMSE (<=99.5th pct): $3,664,002,264,072
```

The model does a good job predicting most home prices. The average error of about $13 million is mostly driven by a few very expensive properties that throw off the scale a bit. For a majority of the homes, the median error is around $175,000, which is a much better reflection of how well the model performs day to day. Even after trimming out the most extreme cases, the results stay consistent, showing that the model captures real housing price patterns without being thrown off by unusually high or low sales.

9.3.3 Train/Test Performance Summary

Table 4: Performance Summary

Set	Adj. R2	mae (log)	rmse (log)	mae (usd)	rmse (usd)
Train	0.361	0.493	0.684	NA	NA
Test	NA	0.499	0.765	172271	3664002264072

9.4 Summary of Final Model and Model Choice

$\log(\text{price}_i) = \beta_0 + \beta_1 \text{bed}_i + \beta_2 \text{bath}_i + \beta_3 texthouse\_size_i + \beta_4 \text{house\_size}_i^2 + \beta_5 \text{acre\_lot}_i + \beta_6 \text{acre\_lot}_i^2 + \beta_7 (\text{bath} \times \text{house\_size})_i + \epsilon_i$

Based on all diagnostics, Model2_int (`model2_int`) , which has quadratic terms for `house_size` and `acre_lot`, as well as an interaction between bath and `house_size` is chosen as the final model. It outperforms Model2 on all selection criteria, achieving a lower AIC and higher Adjusted R^2, while maintaining interpretability and theoretical consistency.

Summary of selection criteria:

- Functional form: RESET tests indicated non-linearity and functional misspecifications which was initially addressed by adding squared terms (Model2). However, because some non-linearity remained (as seen in the second RESET re-test), the specification was further resolved by including bath:house_size interaction term (Model2_int), leading to the final chosen form.
- AIC/BIC: AIC and BIC dropped dramatically from Model2 to Model2_int(AIC:46,719 to 45,443; BIC 46,783 to 45,514), indicating a vastly superior fit
- Adjusted R^2: Increased from 0.32 (Model2) to 0.361 (Model_int), showing significantly improved explanatory power.
- RESET: The statistic significantly decreased when moving to Model2, and the decision to include the interaction term in Model2_int, further resolves the functional form raised by this test.
- Heteroskedasticity: Detected by BP test, confirming model inefficiency. Robust standard errors (HC1) were applied to Model2_int, ensuring all inferences and coefficient stability are valid despite the non-constant variance.

- Residuals: Show no major patterns, confirming the functional form is largely correct. The persistent funnel shape at the high end is managed by the robust standard errors in Model2_int though some heteroskedasticity remains.

The final model captures key relationships in the data while remaining interpretable and statistically sound.

---

10) Out-of-Sample Testing

The final model from Section 9 (Model2_int, the Quadratic + bath:house_size interaction model) is now tested on the hold-out test set. This step evaluates how well the model predicts unseen data, a key measure of generalizability.

10.1 Prediction and Error Calculation

```
## Mean Absolute Error (log scale): 0.505
```

```
## Root Mean Squared Error (log scale): 0.782
```

Interpretations:

The model's mean absolute error (MAE) of 0.499 and root mean squared error (RMSE) of 0.765 suggest that predictions are reasonably close to actual prices on the log scale.

The MAE measures the average absolute magnitude of error, while the RMSE penalizes larger prediction deviations more heavily. Although the RMSE is slightly higher than the prior model (a trade-off in out-of-sample precision), this is accepted because the current model's superior AIC and Adj. R^2 confirm a more robust and accurate functional form. These values indicate the model performs adequately overall, with larger errors concentrated among some high-priced homes. The outcomes show consistent accuracy and no major signs of over-fitting when applied to test data.

11) Overall Summary and Discussion

The analysis shows that home size, lot area, and the number of bathrooms have the strongest impact on listing prices, but with diminishing returns as properties get larger. The final model confirms that the value added by a bathroom is condition by the effect of the house size (bath:house_size interaction), reinforcing that amenities provide more marginal value in larger properties. The final model fits the data well and predicts most homes accurately, however, it still struggles with very expensive listings. Including more location based details such as neighborhood quality or regional market trends, could further enhance future models. Overall, the model provides a solid, interpretable view of how key housing features shape prices whole leaving room for refinement with richer data.

Appendix

Appendix 0: Setup Chunk

```
#Load libraries
library(tidyverse)
library(GGally)
library(glue)
library(janitor)
library(broom)
library(scales)
library(readr)
library(car)          # for VIF
library(lmtest)      # for BP test
library(sandwich)   # for vcovHC

#Read in the data
df <- read_csv("real_estate_sample.csv")
```

```
## Rows: 44526 Columns: 12
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (3): status, city, state
## dbl  (8): brokered_by, price, bed, bath, acre_lot, street, zip_code, house_size
## date (1): prev_sold_date
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
#Group variables
numeric_vars <- c("price","bed","bath","acre_lot","house_size")
categorical_vars <- c("status","state")

#Set seed
set.seed(104)
```

Appendix 2.3.: Code Used for Summary Table

```
# Simple numeric summary table
# Remove scientific notation
options(scipen = 999)

summary_table <- df %>%
  #Choose only the numeric columns
  select(price, bed, bath, house_size, acre_lot) %>%
  #Stacks all those columns into two columns
  pivot_longer(cols = everything(), names_to = "variable", values_to = "value") %>%
  group_by(variable) %>%
  summarise(
    n = sum(!is.na(value)),
```

```
    mean = mean(value, na.rm = TRUE),
    median = median(value, na.rm = TRUE),
    sd = sd(value, na.rm = TRUE),
    min = min(value, na.rm = TRUE),
    q1 = quantile(value, 0.25, na.rm = TRUE),
    q3 = quantile(value, 0.75, na.rm = TRUE),
    max = max(value, na.rm = TRUE)
  )

knitr::kable(
  summary_table, digits = 2, caption = "Summary Statistics for Numeric Variables"
  )
```

Appendix 2.4: Code Used for Histograms & Dynamic Caption

```
numeric_vars <- c("price","bed","bath","house_size","acre_lot")

for (v in numeric_vars) {
  x <- df[[v]][!is.na(df[[v]])]
  is_discrete <- v %in% c("bed","bath")

  # Decide on transform for wide-range continuous vars (skip for bed/bath)
  if (!is_discrete &&
      max(x, na.rm=TRUE) / min(x[x > 0], na.rm=TRUE) > 1000) {
    x <- log10(x[x > 0])
    main_title <- paste0("Histogram of log10(", v, ")")
    xlab_text  <- paste0("log10(", v, ")")
  } else {
    main_title <- paste("Histogram of", v)
    xlab_text  <- v
  }

  # For discrete vars, truncate at 10 for visualization only
  if (is_discrete) {
    n_trunc <- sum(x > 10, na.rm = TRUE)
    x_plot  <- x[x <= 10]
    cap <- paste0(
      "Histogram of ", v,
      " (truncated at 10; ", n_trunc, " value(s) > 10 not shown). ",
      "Density lines not applicable due to discrete variable."
    )
    # Set caption for THIS plot
    knitr::opts_current$set(fig.cap = cap)

    # Integer bins look better for counts
    brks <- seq(floor(min(x_plot, na.rm=TRUE)),
                ceiling(max(x_plot, na.rm=TRUE)) + 0.5, by = 1)

    hist(
      x_plot, main = main_title, xlab = xlab_text,
      col = "skyblue", border = "white",
```

```
      breaks = brks, probability = TRUE, xlim = c(min(brks), 10)
    )

    # No density lines for discrete variables
  } else {
    # Continuous variables (possibly log-transformed)
    cap <- paste0("Histogram of ", v, " with density overlay.")
    knitr::opts_current$set(fig.cap = cap)

    hist(
      x, main = main_title, xlab = xlab_text,
      col = "skyblue", border = "white",
      breaks = 60, probability = TRUE
    )
    lines(density(x, na.rm=TRUE), col="red", lwd=2)
    legend("topright", legend="Density", col="red", lwd=2, bty="n")
  }
}
```

```
n_bed_out  <- sum(df$bed  > 10, na.rm = TRUE)
n_bath_out <- sum(df$bath > 10, na.rm = TRUE)

txt <- glue::glue(
  "For discrete variables (`bed` and `bath`), density lines not applicable; ",
  "observations exceeding 10 were truncated for plotting purposes ",
  "(n = {n_bed_out} for bedrooms; n = {n_bath_out} for bathrooms). ",
  "These outliers remain in the dataset for analysis."
)

knitr::asis_output(paste0("\n\n*", txt, "*\n\n"))
```

Appendix 2.5: Code Used for Boxplots

```
numeric_vars <- c("price", "bed", "bath", "house_size", "acre_lot")

for (v in numeric_vars) {
  x <- df[[v]][!is.na(df[[v]])]

  # If the variable spans several orders of magnitude, plot on log10 scale for readability
  use_log <- (is.numeric(x) &&
              any(x > 0, na.rm = TRUE) &&
              (max(x, na.rm = TRUE) / max(1, min(x[x > 0], na.rm = TRUE))) > 1000)

  if (use_log) {
    x_plot  <- log10(x[x > 0])  # exclude zeros for log
    mainlab <- paste0("Boxplot of log10(", v, ")")
    xlab    <- paste0("log10(", v, ")")
  } else {
    x_plot  <- x
    mainlab <- paste("Boxplot of", v)
    xlab    <- v
```

```
  }

  boxplot(x_plot,
          horizontal = TRUE,
          col = "lightgray",
          main = mainlab,
          xlab = xlab,
          outline = TRUE)    # keep outliers visible
}
```

Appendix 2.6: Code Used for Correlation Matrix

```
# Find correlations
cor_matrix <- cor(df[numeric_vars], use = "pairwise.complete.obs")

# Display nicely rounded table
knitr::kable(
  round(cor_matrix, 2),
  caption = "Correlation Matrix of Numeric Variables"
)
```

Appendix 2.7: Code Used for Categorical Variable Plots

```
# Frequency table
temp_status <- table(df$status)

# Convert to percent
pct_status <- round(100 * prop.table(temp_status), 1)

# Create bar plot and save bar positions
bp <- barplot(
  pct_status,
  main = "Status Composition (%)",
  col = "lightblue",
  xlab = "Status",
  ylab = "Percent",
  ylim = c(0, max(pct_status) * 1.2)  # add headroom for labels
)

# Add text labels above bars
text(
  x = bp,
  y = pct_status,
  labels = paste0(pct_status, "%"),
  pos = 3,           # position 3 = above the bar
  cex = 0.8,         # smaller font
  col = "black"
)
```

Appendix 2.8: Code Used for State Composition Plot

```r
# Frequency table of states
temp_state <- sort(table(df$state), decreasing = TRUE)

# Top 25 + "Other"
top_n <- 25
top_states  <- temp_state[1:top_n]
other_total <- sum(temp_state[-(1:top_n)])

# Put "Other" first, then top states, and sort by count
temp <- c(Other = other_total, top_states)
ord  <- order(temp, decreasing = TRUE)

# Give more room on the left for long labels (restore old par after)
op <- par(mar = c(4, 9, 4, 2) + 0.1)  # bottom, left, top, right
on.exit(par(op))

barplot(
  temp[ord],
  horiz = TRUE,
  las   = 1,                 # keep labels horizontal
  col   = "lightblue",
  main  = "Top 25 States + Other",
  xlab  = "Count",
  cex.names = 0.85,          # slightly smaller label text
  xaxs  = "i"                # axis starts exactly at 0
)
```

Appendix 2.9: Code Used for Train/Test Split

```r
# 80/20 train-test split
train_indices <- sample(1:nrow(df), size = 0.8 * nrow(df))
train_set <- df[train_indices, ]
test_set  <- df[-train_indices, ]

# Check sizes
nrow(train_set); nrow(test_set)
```

Appendix 2.10: Code Used for Numeric Split Check

```r
# Compare key variable (price)
summary(train_set$price)
summary(test_set$price)

# Quick numeric check
train_mean <- mean(train_set$price, na.rm = TRUE)
test_mean  <- mean(test_set$price, na.rm = TRUE)
train_median <- median(train_set$price, na.rm = TRUE)
```

```r
test_median  <- median(test_set$price, na.rm = TRUE)

cat("Train mean:", round(train_mean, 0),
    " | Test mean:", round(test_mean, 0), "\n")
cat("Train median:", round(train_median, 0),
    " | Test median:", round(test_median, 0))
```

Appendix 2.11: Code Used for Categorical Split Check

```r
# Compare percent of listings by state in each split
round(100 * prop.table(table(train_set$state)), 1)[1:5]
round(100 * prop.table(table(test_set$state)), 1)[1:5]
```

Appendix 3.2: Code Used for Baseline Regression

```r
# Run baseline linear regression
model1 <- lm(log(price) ~ bed + bath + house_size + acre_lot, data = train_set)

# Display summary
summary(model1)
```

Appendix 4.1: Code Used for VIF Check

```r
# Calculate VIF for each predictor
vif_values <- vif(model1)

# Display rounded results
vif_table <- data.frame(
  Variable = names(vif_values),
  VIF = round(vif_values, 2)
)

vif_table
```

Appendix 5.1: Code Used for Model Refinement

```r
# Stepwise selection by AIC
model_refined <- MASS::stepAIC(model1, direction = "both", trace = FALSE)
summary(model_refined)
```

Appendix 5.2: Code Used for Model Comparison

```r
# Compare AIC and BIC between baseline and refined models
cat("Baseline AIC:", AIC(model1), "\n")
cat("Refined AIC:", AIC(model_refined), "\n")
cat("Baseline BIC:", BIC(model1), "\n")
cat("Refined BIC:", BIC(model_refined), "\n")
```

Appendix 6.1: Code for Residual vs Fitted Plot

```r
plot(model1$fitted.values, resid(model1),
     main = "Residuals vs Fitted Values",
     xlab = "Fitted Values",
     ylab = "Residuals",
     col = "skyblue", pch = 16)
abline(h = 0, col = "red", lwd = 2)
```

Appendix 7.1: Code for RESET Test (power = 2)

```r
# Ramsey RESET using powers of regressors (adds squares and 2-way interactions)
p_reset2 <- lmtest::resettest(model1, type = "regressor", power = 2)
p_reset2
```

Appendix 7.2: Code for RESET Test (power = 2:3)

```r
# RESET using default powers (2 and 3)
p_reset23 <- lmtest::resettest(model1, type = "regressor", power = 2:3)
p_reset23
```

```
##
##  RESET test
##
## data:  model1
## RESET = 407.6, df1 = 8, df2 = 21859, p-value < 0.00000000000000022
```

Appendix 7.3: Code for Model Adjustment

```r
# Model with quadratic terms
model2 <- lm(log(price) ~ bed + bath + house_size + I(house_size^2) +
                          acre_lot + I(acre_lot^2),
             data = train_set)

summary(model2)

# Compare fit statistics
cat("Baseline AIC:", AIC(model1), "\n")
cat("Quadratic model AIC:", AIC(model2), "\n")
```

Appendix 7.4: Code for RESET Re-Test After Correction

```
# RESET on improved model
p_reset2_new <- lmtest::resettest(model2, type = "regressor", power = 2)
p_reset2_new
```

Appendix 8.1: Code for Breusch-Pagan Test

```
bp_result <- lmtest::bptest(model2)
bp_result
```

Appendix 8.2: Code for Goldfeld-Quandt Test

```
mf <- model.frame(model2)
lmtest::gqtest(model2, order.by = ~ house_size, data = mf)
```

Appendix 8.3: Code for Correction Using Robust Standard Errors

```
vcov_white <- sandwich::vcovHC(model2, type = "HC1")
lmtest::coeftest(model2, vcov = vcov_white)
```

Appendix 9.1: Code for In-Sample Fit Plot

```
# Predicted log(price) vs actual log(price)
# Use the model's own data to guarantee matching lengths
mf <- model.frame(model2)                       # rows actually used in model2
y_actual  <- model.response(mf)                 # log(price)
y_pred    <- fitted(model2)

plot(y_actual, y_pred,
     main = "Predicted vs Actual (Training Set, Log Scale)",
     xlab = "log(Actual Price)", ylab = "log(Predicted Price)",
     col = "skyblue", pch = 16)
abline(0, 1, col = "red", lwd = 2)
```

Appendix 9.2: Code for Model Performance Metrics

```
cat("Adjusted R²:", round(summary(model2)$adj.r.squared, 3), "\n")
cat("AIC:", round(AIC(model2), 2), "\n")
cat("BIC:", round(BIC(model2), 2), "\n")
```

Appendix 9.2.1: Model comparison: Quadratic vs Interaction

```
model2_int <- lm(
  log(price)~ bed + bath + house_size + I(
    house_size^2) + acre_lot + I(acre_lot^2) + bath:house_size, data = train_set
  )

final_model <- model2_int

cmp <- tibble::tibble(Model = c("Quadratic only", "Quadratic + bath:house_size"),
AIC   = c(AIC(model2), AIC(model2_int)),
BIC   = c(BIC(model2), BIC(model2_int)),
AdjR2 = c(summary(model2)$adj.r.squared, summary(model2_int)$adj.r.squared))

# round only numeric columns
cmp_out <- cmp
num_cols <- sapply(cmp_out, is.numeric)
cmp_out[num_cols] <- lapply(cmp_out[num_cols], round, 3)

knitr::kable(cmp_out, caption = "Model comparison: interaction")
```

Appendix 9.3: Code for Residual Diagnostics Plot

```
plot(fitted(model2), resid(model2),
     main = "Residuals vs Fitted (Training Set)",
     xlab = "Fitted log(Price)",
     ylab = "Residuals (log scale)",
     col = "lightblue", pch = 16,
     xlim = c(10, 16), ylim = c(-3, 3))
abline(h = 0, col = "red", lwd = 2)
```

Appendix 9.3.1: QQ Plot and Cook's Distance Diagnostics

```
# Set consistent plotting style for both subplots
par(
  mfrow = c(1, 2), cex.main = 1.2, cex.lab = 1.1,
  cex.axis = 0.9,  font.main = 2, family = "sans") # uniform font style

# QQ Plot
qqnorm(resid(model2),
       main = "Normal QQ Plot",
       xlab = "Theoretical Quantiles",
       ylab = "Sample Quantiles",
       pch = 19, col = "darkgray")
qqline(resid(model2), col = "red", lwd = 2)

cd <- cooks.distance(model2)
n <- length(model2$fitted.values)
```

```r
thr <- 4 / n

#plot
plot(cooks.distance(model2),
     type = "h",
     col = ifelse(cd > thr, "lightblue", "gray60"),
     main = "Influence of Observations",
     xlab = "Observation Number",
     ylab = "Cook's Distance")

# add threshold line
abline(h= thr, col = "red", lty = 2)

# label top 3 most influential points
top <- order(cd, decreasing = TRUE)[1:min(3, length(cd))]
text(top, cd[top], labels = top, pos = 3, cex = 0.8)

# Reset plot layout
par(mfrow = c(1, 1))
```

Appendix 9.3.2: Out-of-sample errors in dollars

```r
# Ensure same variables and complete cases
vars <- c("price","bed","bath","house_size","acre_lot")
idx <- complete.cases(test_set[, vars])

# Predict log(price)
yhat_log <- predict(final_model, newdata = test_set[idx, ])
y_actual <- test_set$price[idx]

mae <- mean(abs(log(y_actual) - yhat_log))
rmse <- sqrt(mean((log(y_actual) - yhat_log)^2))

# Duan smearing factor (corrects bias from log transform)
res_train <- residuals(final_model)
smear     <- mean(exp(res_train), na.rm = TRUE)

# Back-transform to dollars
yhat <- exp(yhat_log) * smear

# compute MAE and RMSE in dollars
mae_usd <- mean(abs(y_actual - yhat), na.rm = TRUE)
rmse_usd <- sqrt(mean((y_actual - yhat)^2, na.rm = TRUE))

# Present final out-of-sample error metrics in dollar terms
cat("mae ($): ", scales::dollar(round(mae_usd, 0)), "\n",
    "rmse ($): ", scales::dollar(round(rmse_usd, 0)), "\n", sep = "")
```

```
## mae ($): $61,012,903,987
## rmse ($): $4,474,766,265,964
```

```r
# --- Robust Summaries (median and trimmed) ---

# Median absolute error (robust)
medae_usd <- median(abs(y_actual - yhat), na.rm = TRUE)

# Trimmed MAE/RMSE (drop top 0.o% by actual price)
trim_idx <- y_actual < quantile(y_actual, 0.995, na.rm = TRUE)
mae_trim <- mean(abs(y_actual[trim_idx] - yhat[trim_idx]), na.rm = TRUE)
rmse_trim <- sqrt(mean((y_actual[trim_idx] - yhat[trim_idx])^2, na.rm = TRUE))

cat("Median AE ($): ", scales::dollar(round(medae_usd, 0)), "\n",
    "Trimmed MAE (<=99.5th pct): ", scales::dollar(round(mae_trim, 0)), "\n",
    "Trimmed RMSE (<=99.5th pct): ", scales::dollar(round(rmse_trim, 0)), "\n",
    sep = "")
```

```
## Median AE ($): $168,727
## Trimmed MAE (<=99.5th pct): $61,320,666,823
## Trimmed RMSE (<=99.5th pct): $4,486,039,312,910
```

Appendix 9.3.3 Train/Test Performance Summary

```r
### Train/Test Performance Summary

perf_tab <- tibble::tibble(
  Set = c("Train", "Test"),
  AdjR2 = c(summary(final_model)$adj.r.squared, NA_real_), # only for train
  mae_log = c(mean(abs(resid(final_model))), mae),
  rmse_log = c(sqrt(mean(resid(final_model)^2)), rmse),
  `mae (usd)` = c(NA_real_, medae_usd),               # <- use the *object* mae_usd
  `rmse (usd)` = c(NA_real_, rmse_trim)                 # <- use the *object* rmse_usd
)
to_dollar <- function(x) paste0("$", format(round(x, 0), big.mark = ",", trim = TRUE))

perf_tab_round <- perf_tab %>%
    dplyr::mutate(dplyr::across(c(AdjR2, mae_log, rmse_log), ~round(.x, 3)),
  `mae (usd)`  = c(NA_real_, round(medae_usd, 0)),
  `rmse (usd)` = c(NA_real_, round(rmse_trim, 0))
   ) %>%

dplyr::select(Set, `Adj. R2` = AdjR2,
       `mae (log)` = mae_log,
       `rmse (log)` = rmse_log,
       `mae (usd)`,
       `rmse (usd)`
       )

knitr::kable(perf_tab_round, caption = "Performance Summary", align = "lrrrrr")
```

Appendix 10.1: Code for Out-of-Sample Prediction and Error Calculation

```r
# Define the final, best performing model (Model2_int: Quadratic + Interaction)
final_model <- lm(
  log(price) ~ bed + bath + house_size + I(house_size^2),
  + acre_lot + I(acre_lot^2) + bath:house_size,
data = train_set
)

# Ensure complete test observations for prediction
vars <- c("price","bed","bath","house_size","acre_lot")
idx  <- complete.cases(test_set[, vars])
pred_test <- predict(final_model, newdata = test_set[idx, ])

# Calculate errors on log scale
mae <- mean(abs(log(test_set$price[idx]) - pred_test))
rmse <- sqrt(mean((log(test_set$price[idx]) - pred_test)^2))

cat("Mean Absolute Error (log scale):", round(mae, 3), "\n")
cat("Root Mean Squared Error (log scale):", round(rmse, 3), "\n")
```